ST

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30, H04L 29/06, G06F 1/00 — A2

(11) International Publication Number: WO 98/24037

(43) International Publication Date: 4 June 1998 (04.06.98)

(21) International Application Number: PCT/US97/20929

(22) International Filing Date: 17 November 1997 (17.11.97)

(30) Priority Data:

08/756,162	25 November 1996 (25.11.96)	US
08/792,092	31 January 1997 (31.01.97)	US
08/872,082	10 June 1997 (10.06.97)	US
08/911,796	15 August 1997 (15.08.97)	US

(71) Applicant *(for all designated States except US)*: HYPERLOCK TECHNOLOGIES, INC. [US/US]; 8401 N. Crawford, Skokie, IL 60076 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: FENG, Jie [CN/US]; 1210 Manor Drive, Wilmette, IL 60091 (US). MAGES, Kenneth, G. [US/US]; 1671 Stranth Erin, Highland Park, IL 60035 (US).

(74) Agents: GERSTEIN, Milton, S. et al.; Hamman & Benn, Suite 3300, 10 S. LaSalle Street, Chicago, IL 60603 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD OF SECURE SERVER CONTROL OF LOCAL MEDIA VIA A TRIGGER THROUGH A NETWORK FOR INSTANT LOCAL ACCESS OF ENCRYPTED DATA ON LOCAL MEDIA

(57) Abstract

A method of transmitting protected video and/or graphic data over the Internet from a Web site, by encrypting the video and/or graphic data and storing it at a Web site associated with a server, and by encrypting a video player and storing it at the Web site. Both are then downloaded to a requesting computer via the Internet or Intranet. The requesting computer decrypts the video and/or graphic data and video player via a previously supplied decryption key, so that the video may be played back by the decrypted player.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD OF SECURE SERVER CONTROL OF LOCAL MEDIA VIA A TRIGGER THROUGH A NETWORK FOR INSTANT LOCAL ACCESS OF ENCRYPTED DATA ON LOCAL MEDIA

BACKGROUND OF THE INVENTION

The present invention is directed to a method of transmitting "triggering data" over a network to cause video and/or audio information data on a CD-ROM at an end-user's computer to be made readable. In addition, the CD-ROM comprises program files for automatically dialing and connecting the end-user's computer to a targeted host's server using an operating system such as "Windows 95". The CD-ROM will only allow the end-user access to the video/and or audio on it by logging onto the host's server via a network such as the Internet.

The Internet is a conglomeration of computer networks that are linked together. Each network of the Internet may have one or more servers, and an operating system that may be different from that of others in the Internet. To link one

network to another, and in order to overcome these operating differences between computer networks, the Internet system utilizes hardware and software devices called: bridges, routers, and gateways, all of which adapt the information being sent on one network to the operating and protocol requirements of the receiving network. For example, a gateway will connect, or "splice" a network operating on the Novell protocol to a network that operates on a DECnet or SNA protocol.

There are currently more than 10,000 computer networks that are linked together, worldwide, which together constitute the "Internet". Because they do not all operate on the same operating system, and because of different protocols, the data sent from one host computer of one network to a receiving computer of another network - which may be many thousands of miles away from the host computer - may take a relatively long time, since the gateways, bridges and routers must conform or adapt the protocol of the sending host computer to the receiving computer's protocol.

In addition to the time-delays associated with protocol variances, the Internet when connecting to an end-user via Plain Old Telephone Service (POTS), has a maximum data-transmission capacity of 3.6 kbytes per second, which is not enough for sending video images in real time.

The Internet system utilizes two types of file-transfer protocols (FTP) for copying a file from a host computer to the receiving computer: ASCII and binary. An ASCII file is a text file, while every other kind of file is binary. ASCII

files are transmitted in seven-digit ASCII codes, while the binary files are transmitted in binary code. Because all data stored in computer memory is stored in binary format, when one sends a file in the Internet, it is sent in binary format. However, as discussed above, owing to the data-transmission constraints imposed by the Internet system because of the differing operating systems, and a multitude of gateways, routers, and bridges, the file data must be sent out in packets of a size no greater than 1536 bytes. Since the size of just a thirty-second video may be as great as 2.5 megabytes, it may take up to one-half hour or more to send a thirty-second video over the Internet from a host computer to a receiving computer. Presently, there are compression techniques that compress the files in order to reduce this play-back-time, which data is decompressed at the receiving computer. An example of such a system is VDOLive, manufactured by VDOnet Corp. of Santa Clara, California. However, these compression-systems still send the data in binary format, requiring packet-data sizes of no greater than 1536 bytes. Thus, even with these compression-systems, the length of time to receive a thirty-second video over the Internet after being buffered in the user's computer is near real time, but is unstable, choppy and drops as much as 96% of the video data over a conventional phone line.

In the Internet, there is an electronic-mail delivery system called E-mail. The E-mail system utilizes addresses to direct a message to the recipient, with each address having a mailbox code and a daemon, with the mail box and daemon being

separated by the symbol @. In the E-mail delivery system, all of the messages or "mail" are routed through selected routers and gateways, until it reaches what may be called a "post office" that services the recipient to whom the electronic mail is to be delivered. The "post office" is a local server. The need for these local "post offices" is because there is every reason to assume that the recipient-computer, to which the mail is being sent, is either not powered up, or is performing a different task. Since most computers in the Internet are not multi-tasking machines, such as, for example, computers running on the DOS operating system, if such a computer be engaged in performing a task, it is not possible for it to receive the E-mail data at that time. Thus, the local "post office" or server stores the message until such a time as it may be delivered to the end-user to whom it is intended.

In the E-mail system, there has really been only one format standard for Internet messages. A variation has been the MIME version, which stands for Multipurpose Internet Mail Extensions, which defines a new header-field, which is intended for use to send non-text messages, such as multimedia messages that might include audio or images, by encoding the binary into seven-digit ASCII code. Before MIME, the limitation of E-mail systems was the fact that it would limit the contents of electronic mail messages to relatively short lines of seven-bit ASCII. This has forced users to convert any non-textual data that they may wish to send into seven-bit bytes representable as printable ASCII characters before

invoking a local mail UA (User Agent, a program with which human users send and receive mail). Examples of such encodings currently used in the Internet include pure hexadecimal, uuencoded, the 3-in-4 base 64 scheme specified in RFC 1421, the Andrew Toolkit Representation [ATK], and many others. Even though a user's UA may not have the capability of dealing with the non-textual body part, the user might have some mechanism external to the UA that can extract useful information from the body part. Moreover, it does not allow for the fact that the message may eventually be gatewayed back into an X.400 message handling system (i.e., the X.400 message is "tunneled" through Internet mail), where the non-textual information would definitely become useful again. With MIME, video and/or audio data may be sent using the E-mail system. MIME uses a number of header-fields, such as "Content-Type" header field, which can be used to specify the type and subtype of data in the body of a message and to fully specify the native representation (encoding) of such data; "text" Content-Type value header field, which an be used to represent textual information in a number of character sets and formatted text description languages in a standardized manner; "multi-part" Content-Type value, which can be used to combine several body parts, possibly of differing types of data, into a single message; "application" Content-Type value, which an be used to transmit application data or binary data, and hence, among other users, to implement an electronic mail file transfer service; "message" Content-Type value, for encapsulating another mail message; "image" Cont-

ent-Type value, for transmitting still image (picture) data; "audio" Content-Type value, for transmitting audio or voice data; "video" Content-Type value, for transmitting video or moving image data, possibly with audio as part of the composite video data format; "Content-Transfer-Encoding" header field, which can be used to specify an auxiliary encoding that was applied to the data in order to allow it to pass through mail transport mechanisms which may have data or character set limitations. Two additional header fields may be used to further describe the data in a message body: The "Content-ID" and "Content Description" header fields.

However, there are considerable drawbacks and deficiencies in transmitting video images and/or audio data over the Internet using E-mail's MIME. Firstly, there is often considerable time delays, such that it may take up to ten or more minutes to send a thirty-second video clip over the E-mail system. In times of high-traffic usage, the delay may even be more than ten minutes. Secondly, the video image or audio data cannot be viewed or listened to by the end-user, or recipient, until all of the data of the entire video or audio file has been received by the receiving computer, which, also, adds a considerable time lag to the actual viewing or listening. Thirdly, the end-user or recipient computer must have the necessary E-mail and MIME software for decoding the data. Fourthly, since MIME is an E-mail protocol system, the data is transmitted via the E-mail system, meaning that it is routed through one or more post offices and servers, which delay the transmission of the data, and which require that no

other task be performed by the receiving computer if it is a single-tasking machine, like DOS-operating system machines. Fifthly, like all E-mail deliveries, the requisite E-mail software at the recipient computer must decode the encoded data received, and then cut-and-paste the data into a new file, such as NOTEPAD, which is time-consuming, before the new file is played back by a viewer or player.

While CD-ROMs provide a great amount of data storage, a new disc called DVD-ROM (digital video disk) provides considerably more data storage, reaching data storage capacities of up to 17 GB as compared to 680 MGB for a CD-ROM. This DVD-disc has especial usefulness in the storage of archiving data and in the storage of video data, such as full-length movies. Conventional CD-ROMS do not provide enough storage capacity for full-length movies, and the like. In conjunction with the DVD-ROM disc, is a new envisioned technology called "Zoom-TV", which will prevent the playback of the DVD-ROM without first obtaining permission form a service-provider. This service-provider will send the necessary enabling data to the system playing the DVD-ROM for allowing the data on the DVD-ROM to be played back, for which the user of the DVD-ROM will be billed, whereby a pay-per-view type of system is effected. The user's system for playing the DVD-ROM will call the service-provider via the land-line telephone network, over which the necessary enabling data for playing the DVD-ROM is also transmitted to the user's or requesting system. The pay-per-view DVD system will typically include a DVD-player, which includes a video player such as MPEG-2, a TV or

monitor; and a microprocessor or personal computer. The user will request permission to playback the video on the DVD-ROM by calling up the service provider via the public, switched telephone network, or PSTN.

DVD-ROMs containing full-length movies presently are provided with parental rating controls, which a three-tier format: To wit, a "Kids' Title" playback only, a "Forbid Adult Titles" mode, and a "Play All Titles" mode. Each title of a DVD-ROM is accorded one of a first, general category allowing playback by any of the three modes, a second "Kids" category for playback only in the "Kids' Title" playback mode and which prevents all other titles including adult titles, and a third "Forbid-Adult" category for which only adult titles are prevented from being played but all other titles may be played. For purposes of this application, the first general category, allowing complete playback of all titles, is assigned the equivalent code of "1" in its heading, while the second Kids' titles only playback mode is assigned a code of "2", and the third "Forbid-Adult" category for which only adult titles are prevented from being played having a code of "3" in its header. The DVD player, such as MPEG-2, has corresponding software for detecting the category codes, and software for setting the level of playback, whether it be the first, second or third mode.

In addition to parental control codes, each DVD-ROM also has a country code, with the code representing the country of manufacture of the DVD-ROM. In conjunction with this, each DVD-player has a country code, with the DVD-player's software

preventing play of the DVD-ROM if the country code on the DVD-ROM does not match the country code of the DVD-player. This system is intended to prevent the illegal copying and pirating of the videos on the DVD-ROM.

Cable-TV networks are well-known. These systems utilize a set-top box converter for receiving the signals from the cable-TV provider and playing them back on the TV or monitor. Cable-TV networks also now have units that allow access to the Internet via the cable network, with such units having their own microprocessor for allowing communication with the Internet and for the display of Internet data on the TV or monitor.

SUMMARY OF THE INVENTION

It is the primary objective of the present invention to separate keys and data by providing a CD-ROM having its informational data of video and/or audio that is crippled, which data may only be read after it has been "uncrippled" by receiving "uncrippling" triggering data over the Internet from a server of a host system, so that a company's host computer serving the Internet may transmit the "uncrippling" data over the Internet to an end-user's receiving computer in order to uncripple and, thereby, actuate the CD-ROM, so that the data thereon may be read by the end-user's receiving computer only in volatile memory such as RAM.

It is another objective of the present invention to enable server control of the local media data by providing such a "crippled" CD-ROM with video and/or audio data thereon, whereby content by a company on the Internet may be

better controlled, and whereby in conjunction with the content, video and/or audio playback may be combined with any updated, textual information, such as current price of a product or products, location of a store or stores in the vicinity of the end-user's residence, etc. Specific tracks on the CD-ROM can thereby be controlled by the remote server.

It is another objective of the present invention to provide such a "crippled" CD-ROM with video and/or audio data thereon, whereby the CD-ROM is inherently provided with Internet start-up and connecting program that automatically and directly connects the end-user's computer to the company's or content provider's host server via the Internet, whereby, not only does such facilitate and encourage the connection of the end-user to the content provider's web page, but also provides the content provider with valuable marketing information, such as the physical location of the caller, whereby selected information unique to that caller may be downloaded to him over the Internet, such as name and addresses of stores of the company or advertiser nearest to the caller, etc.

It is another objective of the present invention to provide such video imaging, with or without audio, such that the use of the E-mail system or the Internet system itself is entirely obviated.

It is another objective of the present invention to provide such video imaging, with or without audio, such that the data representing the video and/or audio is accessed off the end-user's CD-ROM, with the transmitted de-crippling

triggering data from the content provider's host server (URL) being a trigger as small as a few bytes.

It is another objective of the present invention to allow by server permission only, the end-user the ability to store said trigger on non-volatile media for permanent ownership of said data.

It is also an objective of the invention to provide a software program in the end-user computer called a "catcher" for catching the trigger data such as the file header, decoding it, and playing the file header data substantially "on the fly", so that the video and/or audio data on the CD-ROM may be played back on the end-user's computer substantially immediately after having received the trigger data.

It is also an objective of the invention to store both the video files and the video player for playing the video files in encrypted form at the Web site associated with a server of the Internet or Intranet, which encrypted video files and video player are downloaded to a requesting computer having the software decryption keys for the encrypted video files and player, whereby the video files are protected from unauthorized playback.

It is also the primary objective of the present invention to provide a method and system for implementing the pay-per-view DVD-ROM system, whereby the enabling data provided to the DVD-player allowing the playback of the DVD-ROM (Hyper-DVD) video data is provided to the DVD-player via the Internet or via the cable-TV system provider.

It is also the primary objective of the present inven-

tion to provide a method and system for playing back DVD-ROMs which system discriminates between DVD-ROM's requiring pay-per-view play, and those that are free and do not require pay-per-view play.

It is also the primary objective of the present invention to provide a method and system for playing back DVD-ROMs which system discriminates between DVD-ROM's requiring pay-per-view play, and those that do not, by the use of a special code for the header of the DVD-ROM indicating a pay-per-view title.

Toward these and other ends, the method of the invention for transmitting the de-crippling triggering data for video and/or audio off a CD-ROM ("HyperCD") over the Internet consists of encoding the data representing critical information of the file keys such as the header of the video/audio files on the CD-ROM, and transmitting that encoded key to the local server of the local web of the Internet serving the caller, or end-user computer. The local server then establishes a point-to-point socket-connection between the transmitting, host computer, and the receiving or end-user computer, thereby obviating the need to send the actual video data over the Internet. When the encoded key is received by the receiving, or end-user, computer, the data is decoded and matched to the video/audio files of the CD-ROM, whereupon, since the data files on the CD-ROM now have an associated and complete header, the data thereof may be read, to thus allow the instant playback of the video-audio data on the CD-ROM.

Since the encoded header data that is sent over the

Internet is a necessity before the end-user may playback the video/audio data from the CD-ROM, the host computer may send along with the encoded data, additional information pertinent to the information contained on the CD-ROM, such as current prices, special offers or deals, locations of local stores or dealers, or any information that the host computer, content provider, would like the end-user to receive.

In order to encourage the end-user to view the video/audio, the CD-ROM is provided with its own Internet dial-up program files for connecting to the host web server, so that very little time and effort is required on the part of the end-user.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be more readily understood with reference to the accompanying drawings, wherein:

Figure 1 is a pictorial representations of the hardware systems and software processes used for carrying out the present invention;

Figure 2 is a block diagram showing the hardware of the end-user's computer used for carrying out the present invention;

Figure 3 is a flow chart at a user's computer for accessing the trigger-data from a web-site;

Figure 4 is a flow chart for the server associated with the Internet for evaluating the trigger-request from the user's computer and for sending the trigger;

Figure 5 is a block diagram showing the socket-to-socket

connection for transmitting the de-crippling, triggering key for causing the display of the video images and/or audio data of a "HyperCD" at the end-user's PC over the Internet from a host computer combined with a targeted URL to a recipient or end-user's computer;

Figure 6 is a block diagram showing the steps for forming on the CD-ROM the encoded video and audio data for use by the end-user recipient computer after having been crippled by removing the header-triggering key sent from the media files;

Figure 7 is a block diagram showing the process of triggering in order to invoke "HyperCD" video and/or audio data at the receiving computer for playback;

Figure 8 is a pictorial representation of the hardware component and software processing involved;

Figure 9 is a flow chart showing the server-side of the Internet with the encrypted files thereat;

Figure 10 is a flow chart showing the "catcher" program of the invention at the ens-user's computer for playing back the receiving data immediately;

Figure 11 is a block diagram of the catcher-program process;

Figure 12 is a block diagram of a modification of the invention where instead of using a CD-ROM, the video and/or other information is downloaded via the Internet from a Web page, which video and/or other information is encrypted with a key, with the user's computer storing the corresponding decryption key therefor; and

Figure 13 is a block diagram of the DVD-ROM player

system of the invention allowing both pay-per-view DVD-ROM play and conventional, non-pay-per-view DVD-ROM play.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the drawings in greater detail, and to Figures 1 and 2 for now, the hardware used to carry out the present invention is shown. All of the hardware is conventional and well-known, and includes an end-user computer 1 having a CD-ROM drive 2 for playing a CD-ROM 3 having stored thereon crippled data 4 that is unreadable without first having received a trigger or uncrippling key 5. The end-user's computer 1 is connected via the Internet 6 to a host-computer server 7 which has stored thereat the uncrippling or triggering key 5 for the information stored on the end-user's CD-ROM 3. The end-user's computer 1 has a display and a CPU 9 and a communication-device, such as a modem 10 for establishing communication with the Internet 6. The computer 1 also has the CD-ROM drive 2, hard-drive 11, RAM 13, and video system 8 including monitor as well as audio system 13.

Referring to Fig. 3, there is shown the flow charts for receiving the uncrippling key. The end-user first submits a request over the Internet for the uncrippling key (block 60). The user then waits for that key (block 62), and if the user is not authorized, the request is denied. If the request is authorized, then the uncrippling key is sent by the server and received by the end-user's computer (block 64), whereupon the end-user's computer directs the uncrippling key into volatile memory such as RAM, not into a RAM-disk to be vis-

ible, but saved in a dynamically allocated data structure in RAM accessible only by the receiving program, combined with crippled data read from the CD-ROM and displays the video/animation (block 68).

Figure 4 shows the process-flow that at the server side. The server conventionally provides the web pages to the Internet users (block 70), and awaits a user-request (block 72). If a request is received from an end-user's computer, the server evaluates the request (block 74) in order to authorize the transfer of the uncrippling key (block 76). If an authorization is granted, then the uncrippling, trigger key is sent (block 78).

Referring now to Figs. 5-7, video images and/or audio are converted from analog to digital and stored in crippled fashion in digitized format (block 10) on CD-ROM 3. The crippling of the CD-ROM is achieved by removing critical information such as the video-audio header, whereupon such video/audio data is rendered unreadable by the end-user's computer. The "HyperCD" 3 is provided with the URL (web page) of the designated host computer, or server, (block 14), such, as for example: http://tekweb.com/hypercd/adver/lotto.html, which may be used on the CD-ROM for the Illinois Instant Lottery video advertising. Such digitized format may be existing computer memory files (block 12) that are already in binary format, or may be original files originated by recording the video and/or audio, as by a camcorder or tape, etc., and converting the analog signals into digital, or binary, code. In the case of originating files, the analog data may

be converted to digital data using an INTEL "Smart Video Reorder Pro", for example. The raw binary data that is stored on the "HyperCD" (block 16) is crippled, so the only way to access the data is a socket-to-socket connection with the server of the web page of the host. By means of the process performed in block 14, the CD-ROM contains a code representing the URL web page of the host computer where the necessary de-crippling key is located. This data on the CD-ROM 3 will automatically call up and connect the end-user's computer to the host computer's server 7 on the Internet, whereby a socket-to-socket connection is made therebetween (block 18). Such an automatic connection is well-known, and will automatically find the end-user's browser, will call the Internet service provider, and pass the necessary links from the CD-ROM to the browser in order to get to the host's web page. Such software is available on the "Windows 95" operating system, such as "ActiveX". The host computer then sends back to the local server serving the end-user's computer the necessary, uncrippling trigger for the specific video/audio data on the end-user's CD-ROM (block 20). From the local server, the data is sent out directly over the Internet to the end-user, and, in particular, to the RAM 12 of the end-user's computer (block 22). In RAM, the trigger (block 22), and the data on the CD-ROM 3 are combined, and played back (block 24), as described above. However, as will be explained hereinbelow, since the key 5 is being sent via Internet 6, the end-user's computer 7 must be equipped with the requisite software which is capable of receiving data from the server 7

and which will ensure that the received encoded key 5 is placed safely in RAM 12, and not allowed to be otherwise saved in hard drive 11 where it may be captured and used in a way not authorized by the server 7.

Referring to Figure 6, at the end-user computer end, the raw analog data of the audio/video is digitized (block 30), as explained above, and stored on CD-ROM 3 by conventional techniques. During the storage of the data on the key or critical information of the media file such as video-audio header associated with the video/audio files will be omitted from storage on the CD-ROM, whereupon the CD-ROM is crippled, or prevented from being read for playing back the video/audio files (block 32). The CD-ROM is provided with software for linking up the host-computer which has the necessary key 5 for uncrippling the video-audio files 4 on the CD-ROM 3, which linking software maps or automatically directs the end-user's computer to the host server via the Internet, such linking software having all of the necessary routing information for directing the Internet connection to the host computer's server and web page (URL) (Block 34). The encoding of the critical information such as "Header" trigger is achieved utilizing any conventional encoding program, such as, for example, RSA by Data Security (block 36). This encoding will create a trigger of a few bytes comprising all of the necessary information to trigger the CD-ROM, and to invoke the video and/or audio data.

Figure 7 shows the steps involved for de-crippling the data on the CD-ROM 3 of a receiving or end-user's computer 1

(block 40). A socket-to-socket connection is made between the host, or sending, computer and the receiving, or end-user's computer by means of the linking software described above installed on the end-user's computer (block 42). The Internet Service Provider (ISP) of the end-user's computer's web of the Internet sends the data to the host computer's server over the Internet, which means that any number of local servers and gateways and routers will have been involved in transmitting the data, until it finally arrives at the server 7 serving the web associated with the host computer (block 42). As soon as this socket-to-socket connection is made, the encoded trigger 5 is sent, at a rate of about 3.6 kbytes a second (block 44). The end-user's computer has a specially-dedicated software program for catching the key, decrypting the key 5 from the server and data from the CD-ROM 3, combining the key and data and playing it back. This catcher is a software program discussed hereinbelow that will direct the incoming key, such as the header, to a random location in RAM 8 such as cache directory, of the computer (block 46) and the key will only be visible to the program. The catcher is necessary, since, if it were not present, it is the "nature" of personal computers to randomly dump data which has not had a specific destination assigned to it. Thus, without the catcher, the incoming data may be strewn into a different directory and/or sub-directories, to, thus, be irretrievably lost. As soon as the encoded key 5 arrives and is stored in RAM by means of the catcher program, a subroutine "player" in the program in the receiving computer begins to decode the

trigger, in order to invoke the correct track of the CD-ROM (block 48), from which the data passes to the audio/video subsystem (8,13, Fig. 2), in order to play the video or audio (block 50). It is noted, and emphasized, that as soon as the key has been decoded, the video and/or audio data is immediately "played" back by the audio/video subsystems (8,13, Fig. 2), bypassing the necessity of having to first store the key, or other trigger, on a hard drive before playback. Referring specifically to Fig. 8, there are shown the server 1, the user computer 2, and the software processes 3 used for transmitting the uncrippling key 4 over a network 5, the combining in RAM 6 of the key 4 and crippled data 7 from the CD-ROM 8, the rendering or displaying of the media data 9 such as video/audio or animation on the display 10 or from the audio system 11, and the storing of the key 4 to non-volatile media 13, such as a hard drive, for permanent ownership of the encrypted CD media.

It is noted that it is possible to "cripple" the video/audio data on the CD-ROM by other means other than deleting the header thereof. For example, the file could be made a hidden file, with the trigger data from the host computer being a command to remove the hidden status. Alternatively, the video/audio file could have a changed extension, with the trigger data from the host computer being a command to change the extension. Moreover, the crippling of the video/audio file may be achieved by the use of ZIP file, with the trigger data from the host computer being a command to UNZIP the data. It is, also, within the scope and purview

of the invention to use a floppy disk for storing the crippled file, as described above, for those applications requiring less disk-memory, with the uncrippling data from the host server being sent to the floppy-disk drive via the catcher program, as described above for uncrippling the data on the floppy-disk. Of course, the crippled file may also be stored on any storage medium, such as the hard drive 11, with the uncrippling data from the host server being sent to the drive for that storage medium via the catcher program, as explained above. The uncrippling data may also be stored directly in a hard drive or EPROM so that the user has permanent access to it whenever he wishes to uncripple the file; that is, if the user wishes to permanently retain the crippled nature of the data on the CD-ROM, or floppy, he may permanently store the downloaded uncrippling data in hard drive in order to temporarily uncripple the data on the CD-ROM or floppy every time that it is used, as long as such access is authorized by the server.

Referring to Figs. 9-11, the above-discussed "catcher" program is shown. Encrypted files, such as the header for the crippled CD-ROM data at an end-user's computer, is stored at a server associated with the Internet (block 100 in Fig.9). This header-trigger or other file is encoded and encrypted in a conventional manner at the server (blocks 102, 104). This encoding will create a header of about 50K or less comprising all of the necessary information necessary to the video and/or audio data on the CD-ROM, as is well-known in the art. Then, the encoded data is sent to the local web

server (block 36) in order to be sent out over the Internet, and then to the end-user computer. When the end-user computer requests that the trigger be downloaded, according to the process described above (block 106 of Fig. 10), the catcher program at the end-user computer receives the partial data or trigger, such as a header for the CD-ROM file (block 108). The catcher program decodes the data, using a conventional decoder (block 110), and then sends the data directly the conventional player of end-user computer (block 112) for substantially immediate playback. As soon as the encoded header arrives and stored in the cache directory, the program entitled "player" in the receiving computer begins to decode the data, in order to re-generate the original binary code, from which the data passes to a conventional digital-to-analog converter, in order to play the video or audio. It is noted, and emphasized, that as soon as the header has been decoded, the video and/or audio data starts to play back by the digital-to-analog converter. That is, it is not necessary to store the trigger data on a hard drive, although it is possible to do so, if it is desired to allow the end-user unobstructed access to the video or audio files on the CD-ROM, or the like, at any time in the future.

Referring to Fig. 12, an alternative embodiment is shown. In this embodiment, the use of a hyperCD is obviated, and the video and/or audio, and other data, is downloaded via the Internet from a Web page (block 150). The video and/or audio, and other data, are encrypted with an encryption key. Each user who is to be able to access the data at that Web

page will have a corresponding decryption key (block 152) for decrypting the data. In addition to the video or graphic or other data being sent, the Web site will also download the video player, such as JPEG, "QUICKTIME", or the like, to the user's computer via the Internet. The player, such as JPEG, is also encrypted, so that even after the end user has received the video and other data from the Web site via the Internet, the conventional player stored on the user's computer (block 154) will not be able to play the video. The data emanating from the Internet is first identified with the requesting file of the user's computer (block 158), and then sent to the media player for playback (block 160) using the encrypted player, downloaded from the Web site. The encrypted player, such as JPEG, is decrypted, like the video data, using the decryption key (block 152) provided by the provider of the Web site. It is noted that before the video is downloaded from the Web site via the Internet, the user must first enter his password or other protective feature. According to this embodiment of the invention, videos at a Web site are protected from being viewed without proper authorization, and if the downloaded video were stored in memory of the user's computer, it would not be playable without first downloading the encrypted player, such as JPEG, from the Web site. Thus, the Web provider is able to protect his video and/or graphic data from being copied by the end user's computer. Although the end user may be able to print out a graphic, this would be of very poor quality. It is also within scope and purview of the invention to download only

the encrypted player, for playing back encrypted video and/or graphics already stored on the requesting, end user's computer. In this case, the video data may be supplied to the end user in other forms besides the Internet or Intranet, but still may not be played back without use of the encrypted player downloaded from the Web site and then decrypted by the decrypting key at the end user's computer. Alternatively, the encrypted player may be provided to the end user, and only the encrypted video files may be sent over the Internet or Intranet.

Referring now to Fig. 13, a DVD-ROM disk 10 contains a full-length movie, play, special event, and the like. For playing the DVD-ROM, there is provided a DVD-ROM player 12, such as MPEG-2 for playing the video on a TV or monitor 14. Associated with the player is a microprocessor or CPU 16, such as that forming part of a PC, or a dedicated microprocessor. The microprocessor 16 conventionally communicates with the DVD-player 12 via data ports 18. Associated with the microprocessor is memory storage 20 for storing software that allows the system of the invention to discriminate between DVD-ROM's requiring pay-per-view play, and those that are free and do not require pay-per-view play. Specifically, when the DVD-ROM to be played is provided with one of the three parental codes, then the software of the invention will treat that DVD-ROM in the conventional manner, by allowing instant playing thereof. Referring to Fig. 1, this is seen by the software determining that a non-pay-per-view DVD-ROM is present, or non-Hyper-DVD disk, and will automatically pro-

vide a trigger-signal 24 to a data switch 26. The data switch, upon receiving the trigger-signal, will connect a conventional decryption chip 28 to the DVD-player 12, whereupon the data on the DVD-ROM is decrypted and played back, in the conventional manner.

If the software of the invention has determined that the DVD-ROM 10 is a Hyper-DVD, that is, a pay-per-view DVD, by detection of a code 4 rather than one of the three parental codes, via the header extension or binary code on the DVD-ROM, then the communications-portion 30 of the software of the invention will seek to retrieve the enabling data from a service provider by calling the service-provider over the PSTN. According to the invention, this enabling data may be obtained from the Internet, or, alternatively, via a cable company service provider for those users having cable TV service. In the case of obtaining the enabling data from a cable-TV company, the standard cable-box or set-tip box converter 32 is used for the communications. Also, for those users who utilize a cable box having Internet accessing device, the microprocessor 16 may be that microprocessor of the Internet accessing device itself, with the enabling data being transmitted from the Internet or from the cable TV provider. Instead of using a fourth parental code 4 for indicating the presence of a Hyper-DVD-ROM, a separate and distinct country code may be used, which country code, instead of representing an actual country, represents the a Hyper-DVD. The enabling data for allowing access to the DVD-ROM data may be any of those set forth in Applicants' above-

mentioned copending patent applications, such as missing header, etc., and may also include conventional password, ID, security methods, or other standard verification keys, which are well-known and conventional.

After the user's software requests the downloading of the enabling data, the service provider will either send the data, of the requester is a valid customer and current on his account, or will reject the request. If the service provider transmits the necessary enabling data, then the software portion 30 of the invention sends the trigger-data 24 to the data switch 26 to connect the decryption chip 28 to the DVD-player 12.

While the invention is preferably suited for DVD-ROM disks, other large-storage disks, such as laser disks, video disks, etc., may embody the invention. Also, the invention may be used for those DVD-ROMs that do not employ parental and/or country codes; in this case, the code on the DVD-ROM for indicating that it is a Hyper-DVD requiring a verification key or password from a service-provider may be any of those set forth in Applicants' copending applications listed above, such as supplying the missing header, or any other data for uncrippling the crippled data on the DVD-ROM. Also, the use of a password or key, and the like, which would be provided by the service-provider if the requester passes a set of requirements, such as credit check, and the like, may be used.

The following is the software code listing for the server of the host computer's web for bursting the encoded "header" trigger data through the Internet.

SENDFILE.C

```
#!/usr/sbin/perl
# Get the input
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# Split the name-value pairs
@pairs = split(/&/, $buffer);

foreach $pair (@pairs)
{
          ($name, $value) = split(/=/, $pair);

          # UN-Webify plus signs and %-encoding
          $value =~ tr/+/ /;
          $value =~ s/%([a-fA-FO-9][a-fA-FO-9])
          /pack("C", hex($1)) /eg;

          $FORM{$name} = $value;
}
# Location of the CMC files
$CMCDIR = '/UL/people/CMC/' . $FORM{'dir'};

# If the $CMCDIR director is not found, exit
if ( ! -d "$CMCDIR" )
{
          &Error("$CMCDIR not found on this system.
Please check the path and try agai
n\n");
}
# If there are no files in the CMC directory no point trying to
transfer files

else
{
  opendir( THISDIR, "$CMCDIR" );
  @allfiles = grep(/\.CMC/, readdir(THISDIR));
  if( ! @allfiles ) {
          &Error("There are currenly no CMC files
          in this directory.
Try again later."); }
  sort @allfiles;
}

print ("HTTP/1.0 200\n");
print ("Content-type: multipart/x-mixed-replace;boundary=
---ThisRandomString---\n\n");
print ("---ThisRandomString---\n");
```



```
#Send the First file with .IVD extension which invokes
IVIDEO.EXE

print "Content-type: application/x-IVD\N\N";
$CONTENT = `cat $CMCDIR/CMC001.IVD`;
print $CONTENT;
print ("\n---ThisRandomString---\n");


# Now send rest of the .CMC files which would call filehd1.exe
while (@allfiles)
{
          $file = shift @allfiles;
          print "Content-type: application/x-CMC\n\n";
          print "$file\n";
          $CONTENT = `cat $CMCDIR/$file`;
          print $CONTENT;
          print ("\n---ThisRandomString---\n");
}


# Subroutine that tells whats wrong
sub Error
{
          print ("Content-type: texxt/html\n\n");
          print ("<Title>Error</Title>\n");
          print ("<H1>Error: </H1><p>\n");
          print (@_);
          print ("<p><p><hr><a href=\"mailto:cmcinter\
          @suba.com\
          ">Contact webmaster </a>");
          exit ();
}
```

The following is the software code listing at the host-computer for encoding the "header" binary data into seven-digit ASCII text format, and also listed is the software code listing for the "player", or decoder, at each receiving, or end-user, computer, for decoding the encoded text format back into binary:

```
HOOK_MENU1 MENU LOADONCALL MOVEABLE DISCHARDABLE
{
 POPUP "&File"
 {
  MENUITEM "&Encode...", 1169
  MENUITEM "&Decode...", 1170
}

POPUP "&Actions"
{
 MENUITEM "&Concatenate Files...", 1171
 MENUITEM "&View A Report File...", 1172
 MENUITEM "C&lean Directories...", 1173
 MENUITEM SEPARATOR
 MENUITEM "&Display Wincode Task", 1174
 MENUITEM "&Hide Wincode Task", 1175
}

POPUP "&Options"
{
 MENUITEM "&Encode...", 1176
 MENUITEM "&Decode...", 1177
 MENUITEM "&Wincode...", 1178
 MENUITEM "&Winsort...", 1179
 MENUITEM SEPARATOR
 MENUITEM "&Viewer...", 1180
 MENUITEM SEPARATOR
 MENUITEM "&ZIP/UNZIP...", 1181
 MENUITEM SEPARATOR
 MENUITEM "&Hook App...", 1182
}

POPUP "&Help"
{
 MENUITEM "&Contents", 1183
 MENUITEM "&Search for Help on...", 1184
 MENUITEM "&How to Use Help", 1185
 MENUITEM "&Wincode FAQ", 1186
 MENUITEM "C&opyrights", 1187
 MENUITEM SEPARATOR
 MENUITEM "O&rdering the Help file...", 1188
 MENUITEM SEPARATOR
```



```
 MENUITEM "&About Wincode...", 1189
}
 MENUITEM SEPARATOR
 MENUITEM "&UnHook Wincode", 1190
 MENUITEM SEPARATOR

 MENUITEM "E&xit Wincode", 1191
}
HOOK_MENU2 MENU LOADONCALL MOVEABLE DISCARDABLE
{
 POPUP "&File"
{
 MENUITEM "&Encode...", 2269
 MENUITEM "&Decode...", 2270
}

 POPUP "&Actions"
{
 MENUITEM "&Concatenate Files...", 2271
 MENUITEM "&View A Report File...", 2272
 MENUITEM "C&lean Directories...", 2273
 MENUITEM SEPARATOR
 MENUITEM "&Display Wincode Task", 2274
 MENUITEM "&Hide Wincode Task", 2275
}

 POPUP "&Options"
{
 MENUITEM "&Encode...", 2276
 MENUITEM "&Decode...", 2277
 MENUITEM "&Wincode...", 2278
 MENUITEM "&Winsort...", 2279
 MENUITEM SEPARATOR
 MENUITEM "&Viewer...", 2280
 MENUITEM SEPARATOR
 MENUITEM "&ZIP/UNZIP...", 2281
 MENUITEM SEPARATOR
 MENUITEM "&Hook App...", 2282
}

 POPUP "&Help"
{
 MENUITEM "&Contents", 2283
 MENUITEM "&Search for Help on...", 2284
 MENUITEM "&How to Use Help", 2285
 MENUITEM "&Wincode FAQ", 2286
 MENUITEM "C&opyrights", 2287
 MENUITEM SEPARATOR
 MENUITEM "O&rdering the Help file...", 2288
 MENUITEM "&About Wincode...", 2289
}
 MENUITEM SEPARATOR
```

```
 MENUITEM "&UnHook Wincode", 2290
 MENUITEM SEPARATOR
 MENUITEM "E&xit Wincode", 2291
}
HOOK_MENU3 MENU LOADONCALL MOVEABLE DISCARDABLE
{
 POPUP "&File"
{
 MENUITEM "&Encode...", 3369
 MENUITEM "&Decode...", 3370

}

 POPUP "&Actions"
{
 MENUITEM "&Concatenate Files...", 3371
 MENUITEM "&View A Report File...", 3372
 MENUITEM "C&lean Directories...", 3373
 MENUITEM SEPARATOR
 MENUITEM "&Display Wincode Task", 3374
 MENUITEM "Hide Wincode Task", 3375
}

 POPUP "&Options"
{
 MENUITEM "&Encode...", 3376
 MENUITEM "&Decode...", 3377
 MENUITEM "&Wincode...", 3378
 MENUITEM "&Winsort...", 3379
 MENUITEM SEPARATOR
 MENUITEM "&Viewer...", 3380
 MENUITEM SEPARATOR
 MENUITEM "&ZIP/UNZIP...", 3381
 MENUITEM SEPARATOR
 MENUITEM "&Hook App...", 3382
}

 POPUP "&Help"
{
 MENUITEM "&Contents", 3383
 MENUITEM "&Search for Help on...", 3384
 MENUITEM "&How to Use Help", 3385
 MENUITEM "&Wincode FAQ", 3386
 MENUITEM "C&opyrights", 3387
 MENUITEM SEPARATOR
 MENUITEM "O&rdering the Help file...", 3388
 MENUITEM SEPARATOR
 MENUITEM "&About Wincode...", 3389
}

 MENUITEM SEPARATOR
 MENUITEM "&UnHook Wincode", 3390
 MENUITEM SEPARATOR
 MENUITEM "E&xit Wincode", 3391
}
```



```
HOOK_WORKING DIALOG LOADONCALL MOVEABLE
DISCARDABLE 100, 89, 141, 55
STYLE WS_POPUP  | WS_VISIBLE | WS_CAPTION
CAPTION "Wincode Working..."
FONT 8, "MS Sans Serif"
 LTEXT "", 103, 81, 19, 27, 8
 LTEXT "", 102, 81, 9, 27, 8
 PUSHBUTTON "&Stop", 104, 18, 37, 45, 13
 PUSHBUTTON "&Quit", 105, 78, 37, 45, 13
 RTEXT "Total Job:", -1, 12, 19, 66, 8
 CONTROL "", -1, "STATIC", SS_BLACKFRAME
| WS_CHILD | WS_VISIBLE, 6, 6, 129, 25
 RTEXT "", 101, 12, 9, 66, 8
BASE64_TYPE DIALOG LOADONCALL MOVEABLE
DISCARDABLE 71, 26, 123, 181
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", 1, 12, 163, 45, 13
 RADIOBUTTON "Application: &Octet-Stream:, 301, 12, 19, 99, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
 RADIOBUTTON "Application: &Postscript:, 302, 12, 34, 99, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
 RADIOBUTTON "Image: &JPEG", 303, 12, 49, 99, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
 RADIOBUTTON "Image: &GIF", 304, 12, 64, 99, 12,
     BS_AUTORADIOBUTTON | WS TABSTOP
 RADIOBUTTON "Image: &X-BMP", 305, 12, 79, 99, 12
     BS-AUTORADIOBUTTON | WS_TABSTOP
 RADIOBUTTON "Video: &MPEG", 306, 12, 94, 99, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
 RADIOBUTTON "Audio: X-&WAV", 307, 12, 109, 99, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
 PUSHBUTTON  "Cancel",  2,  66,  163,  45, 13
 GROUPBOX "Content-Type", 101, 6, 5, 111, 152,
     BS-GROUPBOX | WS_GROUP
}
DESC_TEXT DIALOG LOADONCALL MOVEABLE DISCARDABLE 9,50,288,138
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU'
CAPTION "Descriptive Text will be added to first
Encoded file..."
FONT 8, "MS Sans Serif"
{
EDITTEXT  201,  6,  6,  276,  108, ES_MULTILINE | ES_AUTOVSCROLL
      | ES_WANTRETURN
      | WS_BORDER | WS_VSCROLL | WS_TABSTOP
DEFPUSHBUTTON "OK", 1, 69, 120, 60, 13
PUSHBUTTON "Cancel", 2, 159, 120, 60, 13
}
DIR_SELECT DIALOG LOADONCALL MOVEABLE DISCARDABLE 15, 20,
147, 116
STYLE DS_MODALFRAME | WS_OVERLAPPED | WS_CAPTION |
WS_SYSMENUFONT 8, "Helv"
{
EDITTEXT 101, 42, 5, 98, 12, ES_AUTOHSCROLL | WS_BORDER
```

```
| WS_TABSTOP
DEFPUSHBUTTON "OK", 1, 88, 22, 50, 14LISTBOX 103, 6, 30, 64,
82, LBS_STANDARD | WS_TABSTOPPUSHBUTTON "Cancel", 2, 88, 41,
50, 14LTEXT "D&irectories:", -1, 6, 18, 64 10LTEXT
"&Directory:" -1, 6, 6, 36, 10}EXISTS DIALOG LOADONCALL
MOVEABLE DISCARDABLE 41, 34, 177, 54STYLE DS_MODALFRAME |
WS_POPUP | WS_CAPTION | WS_SYSMENUCAPTION "Wincode - Output
File"FONT 8, "MS Sans Serif"{ PUSHBUTTON "&Overwrite", 1, 9,
36, 45, 13 PUSHBUTTON "&Rename", 101, 66, 36, 45, 13
PUSHBUTTON "&Skip File", 2, 123, 36, 45, 13 CTEXT "", 102,
21, 15, 135, 8 CONTROL "", "STATIC", SS_BLACKFRAME | WS_CHILD
| WS_VISIBLE, 15, 6, 147, 21}FILE OPEN DIALOG LOADONCALL
MOVEABLE DISCARDABLE 40, 20, 202, 130STYLE DS_MODALFRAME |
WS_OVERLAPPED | WS_CAPTION | WS_SYSMENUFONT 8, "Helv"{
EDITEXT 100, 42, 6, 98, 12, ES_AUTOHSCROLL | WS_BORDER |
WS_TABSTOP DEFPUSHBUTTON "OK", 1, 146, 5, 50, 14 LISTBOX 102,
6, 44, 64, 82, LBS_STANDARD | WS_TABSTOP LISTBOX 103, 76, 44,
64, 82, LBS_STANDARD | WS_TABSTOP PUSHBUTTON "Cancel", 2,
146, 23, 50, 14 LTEXT "File&name:", -1, 6, 8, 36, 10 LTEXT
"Directory:", -1, 6, 20, 36, 10 LTEXT "", 101, 42, 20, 98, 10
LTEXT "&Files:", -1, 6, 32, 64, 10 LTEXT "&Directories:", -1,
76, 32, 64, 10}RENAME DIALOG LOADONCALL MOVEABLE DISCARDABLE
34, 31, 199, 57STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION |
WS_SYSMENUFONT 8, "MS Sans Serif"{ EDITEXT 102, 6, 21, 171,
12 ES_AUTOHSCROLL | WS_BORDER | WS_TABSTOP PUSHBUTTON "?",
103, 180, 20, 12, 13 DEFPUSHBUTTON "OK", 1, 42, 39, 45, 13
PUSHBUTTON "Cancel", 2, 111, 39, 45, 13 LTEXT "Enter a VALID
DOS filename:", 104, 6, 6, 159, 9}VIEW_RPT DIALOG LOADONCALL
MOVEABLE DISCARDABLE 20, 43, 300, 154STYLE DS-MODALFRAME |
WS_POPUP | WS_CAPTION | WS_SYSMENUCAPTION "Wincode - Report
File Viewer"FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 2, 111, 135, 78, 13
EDITTEXT 101, 6, 15, 288, 99, ES_MULTILINE | ES_READONLY
| WS_BORDER | WS_VSCROLL | WS_HSCROLL | WS_TABSTOPCHECKBOX
"&Delete Report File After Viewing", 103, 6, 117, 138, 12,
BS-AUTOCHECKBOX | WS_TABSTOP LTEXT "File:", -1, 7, 5, 15, 8
LTEXT "", 102, 25, 5, 270, 8}
```


The following is the software code listing at each receiving, or end-user, computer, for the catcher for receiving the uncrippling data in the cache directory of RAM and directing it to the proper drive:

```
MAIN_MENU MENU LOADONCALL MOVEABLE DISCARDABLE
{
POPUP"&File"
{
MENUITEM "&Encode...", 101
MENUITEM "&Decode...", 102
MENUITEM SEPARATOR
MENUITEM "E&xit", 1
}

POPUP "&Actions"
{
MENUITEM "&Concatenate Files...", 103
MENUITEM "&View a Report File...", 104
MENUITEM "C&lean Directories...", 105
MENUITEM SEPARATOR
MENUITEM "&Interactive Drag/Drop", 121
MENUITEM SEPARATOR
MENUITEM "Hook Wincode", 122
}

POPUP "&Options"
{
MENUITEM "&Encode...", 106
MENUITEM "&Decode...", 107
MENUITEM "&Wincode...", 108
MENUITEM "W&insort...", 109
MENUITEM SEPARATOR
MENUITEM "&Viewer...", 110
MENUITEM SEPARATOR
MENUITEM "&ZIP/UNZIP...", 111
MENUITEM SEPARATOR
MENUITEM "&Hook App...", 112
}

POPUP "&Help"
{
MENUITEM "&Contents", 113


MENUITEM "&Search for Help on...", 114
MENUITEM "&How to Use Help", 115
MENUITEM "&Wincode FAQ", 116
MENUITEM "C&opyrights", 117
MENUITEM SEPARATOR
MENUITEM "O&rdering the Help file...", 118
```

```
MENUITEM SEPARATOR
MENUITEM "&About Wincode...", 119
}
}
ABOUT DIALOG LOADONCALL MOVEABLE DISCARDABLE 76, 55, 135, 141
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About CMCCODE"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 2, 14, 123, 45, 13
PUSHBUTTON "More...", 1, 74, 123, 45, 13
CTEXT "CMCCODE:", -1, 45, 9, 45, 8
CTEXT "Video Encoder/Decoder", -1, 10, 18, 114, 8
CTEXT "for the Internet", -1, 34, 27, 66, 8
CTEXT "Copyright\xA9 1993,1994", -1, 24, 72, 87, 8
CTEXT "Snappy Inc.", -1, 44, 63, 45, 8
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 123, 111
CTEXT "Version 1.0", -1, 40, 37, 54, 8
CTEXT "Developers Kit Provided by:", -1, 17, 49, 101, 8
CTEXT "created by Caesar Collazo", -1, 18, 82, 99, 8
CTEXT "cmcinter@suba.com", -1, 12, 103, 111, 8
CTEXT "Questions...Comments...e-mail to:", -1, 9, 93, 117, 8
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 12, 47, 111, 1
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
S_VISIBLE, 12, 59, 111, 1
}
ALL_ONE DIALOG LOADONCALL MOVEABLE DISCARDABLE 35, 31, 132, 60
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "CMCCODE - Encode Filename"
FONT 8, "MS Sans Serif"
{
EDITTEXT 101, 28, 23, 75, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP

DEFPUSHBUTTON "OK", 1, 12, 42, 45, 13
PUSHBUTTON "Cancel", 2, 75, 42, 45, 13
CTEXT "Enter a filename for ALL the files:", -1, 6, 7, 120, 9
}
BASE64_MODE DIALOG LOADONCALL MOVEABLE DISCARDABLE 93,
54, 111, 69
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "BASE64 Method"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 6, 51, 45, 13
RADIOBUTTON "&MIME Conformant", 323, 12, 10, 87, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "&Raw BASE 64", 324, 12, 25, 87, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 60, 51, 45, 13
GROUPBOX "", 106, 6, 2, 99, 42, BS_GROUPBOX
}
CHOOSE_V DIALOG LOADONCALL MOVEABLE DISCARDABLE 15, 20, 174, 78
```



```
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Select a Report File Viewer"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 36, 60, 45, 13
RADIOBUTTON "&Wincode Internal File Viewer (32K Max.)",
     701, 12, 10, 150, 12, BS_AUTORADIOBUTTON |
     WS_GROUP | WS_TABSTOP
RADIOBUTTON "Windows &Notepad", 702, 12, 24, 150, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
RADIOBUTTON "&Other:", 703, 12, 38, 33, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 704, 48, 38, 102, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 705, 153, 38, 12, 13
PUSHBUTTON "Cancel", 2, 93, 60, 45, 13
GROUPBOX "", 101, 6, 2, 162, 54, BS_GROUPBOX
}
CLEAN_DIR DIALOG LOADONCALL MOVEABLE DISCARDABLE 52, 51, 228, 162
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Clean Directories"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 92, 143, 45, 13
CHECKBOX "", 601, 12, 19, 192, 12, BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "?", 605, 207, 19, 12, 13


CHECKBOX "", 602, 12, 34, 192, 12, BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "?", 606, 207, 34, 12, 13
CHECKBOX "", 603, 12, 49, 192, 12, BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "?", 607, 207, 49, 12, 13
CHECKBOX "", 604, 12, 64, 192, 12, BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "?", 608, 207, 64, 12, 13
CHECKBOX "Empty the &Clipboard (release global memory)",
     612, 12, 102, 192, 12, BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "?", 613, 207, 102, 12, 13
PUSHBUTTON "Clean &All Directories", 614, 12, 120, 96, 13
PUSHBUTTON "&Report Files Only (*.rpt)", 615, 120, 120, 96, 13
PUSHBUTTON "Cancel", 2, 165, 143, 45, 13
PUSHBUTTON "&Help", 611, 19, 143, 45, 13
GROUPBOX "Select Directories to Clean", 101, 7, 5, 216, 93,
     BS_GROUPBOX
LTEXT "Status:", -1, 12, 83, 27, 8
LTEXT "", 610, 42, 83, 177, 8
}
DEC_CONFIG DIALOG LOADONCALL MOVEABLE DISCARDABLE 26, 26,
250, 147
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Decode Options"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 195, 9, 45, 13
CHECKBOX "Du&mp Files", 301, 12, 9, 69, 12,
```

```
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&Error Checking", 303, 87, 9, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "Sort b&y Extension", 304, 87, 21, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "E&xtension(s)...", 305, 6, 42, 66, 13
COMBOBOX 306, 120, 41, 42, 60, CBS_DROPDOWNLIST |
     WS_VSCROLL | WS_TABSTOP
EDITTEXT 307, 12, 70, 132, 9, ES_AUTOHSCROLL |
     NOT WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 308, 147, 67, 12, 13
RADIOBUTTON "Def&ault to location of Input file",
     309, 12, 99, 132, 12, BS_AUTORADIOBUTTON |
     WS_GROUP | WS_TABSTOP
RADIOBUTTON "User select &on Decode", 310, 12, 112, 132, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
RADIOBUTTON "&Set:", 311, 12, 125, 27, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 312, 42, 125, 102, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 313, 147, 124, 12, 13
RADIOBUTTON "&Wincode select", 314, 174, 110, 66, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "&User select", 315, 174, 125, 66, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 195, 27, 45, 13

PUSHBUTTON "&Defaults", 316, 195, 45, 45, 13
PUSHBUTTON "&Help", 317, 195, 63, 45, 13
GROUPBOX "Decoded File Name", 102, 168, 96, 75, 45,
     BS_GROUPBOX
LTEXT "Code Type:", -1, 78, 44, 39, 8
GROUPBOX "Decoded File Directory", 101, 6, 87, 156, 54,
     BS_GROUPBOX
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 156, 30
GROUPBOX "Temp Directory", 103, 6, 59, 156, 24, BS_GROUPBOX
}
DEC_EXT DIALOG LOADONCALL MOVEABLE DISCARDABLE 49, 30, 144, 133
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Decode File Extension(s)"
FONT 8, "MS Sans Serif"
{
EDITTEXT 318, 12, 25, 45, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
PUSHBUTTON "&Add", 320, 12, 43, 45, 13
PUSHBUTTON "&Delete", 321, 12, 61, 45, 13
PUSHBUTTON "A&ssociate", 322, 12, 79, 45, 13
LISTBOX 319, 73, 26, 58, 69, LBS_NOTIFY |
     WS_BORDER | WS_BORDER | WS_VSCROLL
DEFPUSHBUTTON "OK", 1, 18, 115, 45, 13
PUSHBUTTON "Cancel", 2, 81, 115, 45, 13
LTEXT "Enter Decode Extension: (Max + 20)", -1, 12, 13, 120, 8
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 132, 102
```



```
LTEXT "Ext. Count:", -1, 73, 95, 39, 8
LTEXT "", 323, 114, 95, 16, 8
}
DEL_FILES DIALOG LOADONCALL MOVEABLE DISCARDABLE 63,20,78,127
STYLE DS_MODALFRAME | WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU
FONT 8, "Helv"
{
DEFPUSHBUTTON "OK", 2, 16, 108, 45, 13
LISTBOX 609, 7, 19, 64, 82, LBS_STANDARD | WS_TABSTOP
CTEXT "Files being deleted:", -1, 4, 7, 69, 10
}
DIR_SELECT DIALOG LOADONCALL MOVEABLE DISCARDABLE 15,20,147,116
STYLE DS_MODALFRAME | WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU
FONT 8, "Helv"
{
EDITTEXT 101, 42, 5, 98, 12, ES_AUTOHSCROLL | WS_BORDER
     | WS_TABSTOP

DEFPUSHBUTTON "OK", 1, 88, 22, 50, 14
LISTBOX 103, 6, 30, 64, 82, LBS_STANDARD | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 88, 41, 50, 14
LTEXT "D&irectories:", -1, 6, 18, 64, 10
LTEXT "&Directory:", -1, 6, 6, 36, 10
}
DONE DIALOG LOADONCALL MOVEABLE DISCARDABLE 21, 32, 207, 54
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "CMCCODE - Done!"
FONT 8, "MS Sans Serif"
{
CTEXT "", 101, 12, 14, 184, 9
CONTROL "", -1, "STATIC", SS_BLACKFRAME |
     WS_CHILD | WS_VISIBLE, 6, 6, 195, 25
DEFPUSHBUTTON "OK", 2, 64, 36, 78, 13
}
DONE_SHOW DIALOG LOADONCALL MOVEABLE DISCARDABLE 21, 32, 207, 54
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "CMCCODE - Done!"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 2, 18, 36, 78, 13
PUSHBUTTON "&View Report File", 1, 111,36, 78, 13
CTEXT "", 101, 12, 14, 184, 9
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 195, 25
}
DRAGDROP DIALOG LOADONCALL MOVEABLE DISCARDABLE 119,85,139,110
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Interactive Drag & Drop"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 18, 92, 45, 13
RADIOBUTTON "&Encode", 802, 13, 39, 48, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "&Decode", 803, 13, 53, 48, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
```

```
RADIOBUTTON "E&xt. Based", 804, 13, 67, 48, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
CHECKBOX "&Zip First", 805, 75, 39, 54, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "U&NZIP After", 806, 75, 53, 54, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&Winsort First", 807, 75, 67, 54, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 75, 92, 45, 13
GROUPBOX "Drop to:", 101, 7, 26, 57, 57, BS_GROUPBOX
GROUPBOX "Options:", 102, 70, 26, 63, 57, BS_GROUPBOX
COMBOBOX 801, 69, 7, 64, 66, CBS_DROPDOWNLIST |
     WS_VSCROLL | WS_TABSTOP
LTEXT "CMC Method:", -1, 6, 10, 60, 8
}
ENC_CONFIG DIALOG LOADONCALL MOVEABLE DISCARDABLE 10, 23,
262, 189
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Encode Options"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 206, 9, 46, 13
CHECKBOX "&Line CheckSums", 201, 12, 9, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&File CheckSums", 202, 12, 24, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "Fil&e Headers", 203, 12, 39, 72, 12
CHECKBOX "File Desc&ription", 204, 12, 54, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "Des&criptive Name", 205, 12, 69, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "I&nclude Table", 206, 90, 9, 72, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "Make E&MBL Files", 207, 90, 24, 72, 12
CHECKBOX "Sin&gle File", 208, 90, 39, 72, 12
CHECKBOX "All &In One File", 209, 90, 54, 72, 12
CHECKBOX "Number b&y Ext.", 210, 90, 69, 72, 12
EDITTEXT 211, 123, 90, 42, 12
EDITTEXT 212, 123, 107, 42, 12
RADIOBUTTON "Def&ault to location of input file", 213, 12,
     138, 132, 12, BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "User select &on Encode", 214, 12, 152, 132, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
RADIOBUTTON "&Set:", 215, 12, 166, 27, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 216, 42, 166, 105, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 217, 150, 166, 12, 13
COMBOBOX 218, 213, 90, 42, 57, CBS_DROPDOWNLIST |
     WS_VSCROLL | WS_TABSTOP
COMBOBOX 219, 213, 107, 42, 39, CBS_DROPDOWNLIST |
     WS_VSCROLL | WS_TABSTOP
RADIOBUTTON "&Wincode select", 220, 177, 148, 69, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "&User select", 221, 177, 164, 69, 12,
```



```
     BS_AUTORADIOBUTTON | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 206, 27, 46, 13
PUSHBUTTON "&Default", 222, 206, 45, 46, 13
PUSHBUTTON "&Help", 223, 206, 63, 46, 13
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 159, 78
LTEXT "Bytes per File (Lines/File):", 103, 6, 93, 114, 8
LTEXT "Extension for Encoded Files:", 104, 6, 110, 108, 8

GROUPBOX "Encoded File Name", 102, 171, 132, 84, 51,
     BS_GROUPBOX
GROUPBOX "Encoded File Directory", 101, 6, 123, 159, 60,
     BS_GROUPBOX
LTEXT "Code Type:", 105, 171, 93, 39, 8
LTEXT "File Type:", -1, 171, 110, 39, 8
}
EXT_INFO DIALOG LOADONCALL MOVEABLE DISCARDABLE 76, 55, 207, 111
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "More About CMCCODE"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 2, 81, 93, 45, 13
CONTROL "", -1, "STATIC", SS_BLACKFRAME |
     WS_CHILD | WS_VISIBLE, 6, 6, 195, 81
CONTROL "", -1, "STATIC", SS_BLACKFRAME |
     WS_CHILD | WS_VISIBLE, 13, 57, 180, 1
LTEXT "CMCCODE Version:", -1, 15, 12, 72, 8
LTEXT "WCodeDLL Version:", -1, 15, 23, 72, 8
LTEXT "HookDLL Version:", -1, 15, 34, 72, 8
LTEXT "Release Date:", -1, 15, 45, 72, 8
LTEXT "Memory:", -1, 15, 62, 72, 8
LTEXT "System Resources:", -1, 15, 73, 72, 8
LTEXT "", 701, 90, 12, 105, 8
LTEXT "", 702, 90, 23, 105, 8
LTEXT "", 703, 90, 34, 105, 8
LTEXT "", 704, 90, 45, 105, 8
LTEXT "", 705, 90, 62, 105, 8
LTEXT "", 706, 90, 73, 105, 8
}
EXT_INFO DIALOG LOADONCALL MOVEABLE DISCARDABLE 76, 55, 207, 111
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "More About CMCCODE"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 2, 81, 93, 45, 13
CONTROL "", -1, "STATIC", SS_BLACKFRAME |
     WS_CHILD | WS_VISIBLE, 6, 6, 195, 81
CONTROL "", -1, "STATIC", SS_BLACKFRAME |
     WS_CHILD | WS_VISIBLE, 13, 57, 180. 1
LTEXT "CMCCODE  Version:", -1, 15, 12, 72, 8
LTEXT "WCodeDLL Version:", -1, 15, 23, 72, 8
LTEXT "HookDLL Version:", -1, 15, 34, 72, 8
LTEXT "Release Date:", -1, 15, 45, 72, 8

LTEXT "Memory:", -1, 15, 62, 72, 8
```

```
LTEXT "System Resources:", -1, 15, 73, 72, 8
LTEXT "", 701, 90, 12, 105, 8
LTEXT "", 702, 90, 23, 105, 8
LTEXT "", 703, 90, 34, 105, 8
LTEXT "", 704, 90, 45, 105, 8
LTEXT "", 705, 90, 62, 105, 8
LTEXT "", 706, 90, 73, 105, 8
}
FILE_O_ZIP DIALOG LOADONCALL MOVEABLE DISCARDABLE 40,20,202,130
STYLE DS_MODALFRAME | WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU
FONT 8, "Helv"
{
EDITTEXT 100, 42, 6, 98, 12, ES_AUTOHSCROLL | WS_BORDER |
     WS_TABSTOP
DEFPUSHBUTTON "OK", 1, 146, 5, 50, 14
LISTBOX 102, 6, 44, 64, 82, LBS_STANDARD |
     LBS_MULTIPLESEL | LBS_EXTENDEDSEL | WS_TABSTOP
LISTBOX 103, 76, 44, 64, 82, LBS_STANDARD | WS_TABSTOP
PUSHBUTTON "&All Files", 104, 146, 45, 50, 14
PUSHBUTTON ">> &Clipboard", 105, 146, 63, 50, 14
CHECKBOX "&ZIP First", 106, 146, 81, 51, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 146, 23, 50, 14
LTEXT "File&name:", -1, 6, 8, 36, 10
LTEXT "Directory:", -1, 6, 20, 36, 10
LTEXT "", 101, 42, 20, 98, 10
LTEXT "&Files:", -1, 6, 32, 64, 10
LTEXT "&Directories:", -1, 76, 32, 64, 10
PUSHBUTTON "&Options...", 107, 146, 105, 50, 14
}
FILE_OPEN DIALOG LOADONCALL MOVEABLE DISCARDABLE 40,20,202,130
STYLE DS_MODALFRAME | WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU
FONT 8, "Helv"
{
EDITTEXT 100, 42, 6, 98, 12, ES_AUTOHSCROLL | WS_BORDER |
     WS_TABSTOP
DEFPUSHBUTTON "OK", 1, 146, 5, 50, 14
LISTBOX 102, 6, 44, 64, 82, LBS_STANDARD | WS_TABSTOP
LISTBOX 103, 76, 44, 64, 82, LBS_STANDARD | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 146, 23, 50, 14
LTEXT "File&name:", -1, 6, 8, 36, 10

LTEXT "Directory:", -1, 6, 20, 36, 10
LTEXT "", 101, 42, 20, 98, 10
LTEXT "&Files:", -1, 6, 32, 64, 10
LTEXT "&Directories:", -1, 76, 32, 64, 10
}
HEADER_TYPE DIALOG LOADONCALL MOVEABLE DISCARDABLE 93,54,111,81
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Header Type"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 6, 63, 45, 13
RADIOBUTTON "&Wincode Standard", 224, 12, 10, 87, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
```



```
RADIOBUTTON "&MIME Conformant", 225, 12, 25, 87, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 60, 63, 45, 13
GROUPBOX "", 106, 6, 2, 99, 54, BS_GROUPBOX
CHECKBOX "&Guess Content-Type", 226, 12, 40, 87, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
}
HOOK_APP DIALOG LOADONCALL MOVEABLE DISCARDABLE 10,74,277,117
STYLE DS-MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Hook Application Options"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 222, 9, 45, 13
EDITTEXT 901, 75, 6, 135, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
EDITTEXT 902, 75, 24, 120, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 903, 198, 24, 12, 13
CHECKBOX "&Case Sensitive Application Name", 904,80,45,126,12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&Prompt for Application on Hook", 905,80,57,126,12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "H&ide Wincode when Hooked", 906,80,69,126,12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&Auto-Hook Wincode on Startup", 907,80,81,126,12,
     BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "&Tune...", 910, 222, 97, 45, 13
PUSHBUTTON "Cancel", 2, 222, 27, 45, 13
PUSHBUTTON "&Default", 908, 222, 45, 45, 13
PUSHBUTTON "&Help", 909, 222, 63, 45, 13
LTEXT "Application Name:", -1, 7, 10, 66, 8
LTEXT "Application Path:", -1, 7, 27, 66, 8
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 75, 42, 135, 54
ICON "THOOK_ICON", -1, 27, 69, 18, 20

LTEXT "Advanced Options:", -1, 7, 45, 66, 8
LTEXT "If you are having problems Hooking an application,
     try this ---->", -1, 7, 100, 213, 8
}
HOOK_TUNE DIALOG LOADONCALL MOVEABLE DISCARDABLE 81,74,151,96
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Hook Tuning"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 24, 78, 45, 13
CHECKBOX "&Create Window List on Hook", 911,12,11,126,12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&Skip Opening Window", 912,12,25,90,12,
     BS_AUTOCHECKBOX | WS_TABSTOP
EDITTEXT 913, 108, 39, 30, 12
COMBOBOX 914, 108, 54, 30, 39, CBS_DROPDOWNLIST | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 81, 78, 45, 13
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 138, 66
```

```
LTEXT "Set Hook Delay (seconds):", -1, 12, 42, 93, 8
LTEXT "Set Hook Menu Range:", -1, 12, 57, 93, 8
}
MEMORY_SWAP DIALOG LOADONCALL MOVEABLE DISCARDABLE 63,65,132,66
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "CMCCODE - Memory Swap"
FONT 8, "MS Sans Serif"
{
EDITTEXT 101, 37, 30, 28, 12, ES_AUTOHSCROLL | WS_BORDER |
     WS_TABSTOP
DEFPUSHBUTTON "OK", 1, 12, 48, 45, 13
PUSHBUTTON "Cancel", 2, 75, 48, 45, 13
CTEXT "Enter a memory allocation swap", -1, 6, 7, 120, 9
CTEXT "value (range + 256KB to 16MB):", -1, 6, 16, 120, 9
LTEXT "KBytes", -1, 68, 32, 27, 8
}
OP_TOOLBAR DIALOG LOADONCALL MOVEABLE DISCARDABLE 102, 57,
     104, 112
STYLE WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Options Toolbar"
FONT 8, "MS Sans Serif"
{
PUSHBUTTON "&Encode...", 1001, -1, 0, 105, 14
PUSHBUTTON "&Decode...", 1002, -1, 14, 105, 14
PUSHBUTTON "&Wincode...", 1003, -1, 28, 105, 14


PUSHBUTTON "W&insort...", 1004, -1, 42, 105, 14
PUSHBUTTON "&Viewer...", 1005, -1, 56, 105, 14
PUSHBUTTON "&ZIP/UNZIP...", 1006, -1, 70, 105, 14
PUSHBUTTON "&Hook App...", 1007, 84, 105, 14
PUSHBUTTON "E&xit Toolbar", 2, -1, 98, 105, 14
}
ORDER_HELP DIALOG LOADONCALL MOVEABLE DISCARDABLE
     61, 21, 228, 258
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Ordering the Help file"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "THANKS!", 2, 77, 240, 75, 13
LTEXT "To order the Wincode Help file, send $5.00
     (U.S. Dollars) to: -1, 13, 9, 204, 8
CTEXT "CMC Interactive\xAE", -1, 58, 21, 114, 8
CTEXT "8 S. Michigan Ave.", -1, 58, 29, 114, 8
CTEXT "Suite 2003", -1, 58, 37, 114, 8
CTEXT "Chicago, IL 60606", -1, 58, 45, 114, 8
LTEXT "This price and address are guaranteed until 6/1/95.
     If you", -1, 13, 57, 204, 8
LTEXT "wish to obtain the Help file after this date,
     please e-mail", -1, 13, 65, 204, 8
LTEXT "first for updated information. Make checks payable to:",
     -1, 13, 73, 204, 8
CTEXT "CMC Interactive", -1, 13, 83, 204, 8
LTEXT "By ordering Help, you obtain the following:",
     -1, 13, 112, 204, 8
```



```
LTEXT " 1)The most recent version of Wincode with the Help
file",    -1, 13, 122, 204, 8
LTEXT " 2)Directly e-mailed pre-releases of future
versions of",  -1, 13, 130, 204, 8
LTEXT "     Wincode and the Help file", -1,13,138,204,8
LTEXT " 3)E-mail (only) technical support", -1,13,146,204,8
LTEXT "All files will be ELECTRONICALLY MAILED to you.
If you",
     -1, 13, 162, 204, 8
LTEXT "wish to have something sent through the US
Postal service,",
     -1, 13, 170, 204, 8
LTEXT "please include a Self-Addressed-STAMPED Disk Mailer AND",
     -1, 13, 178, 204, 8
LTEXT "Disk with your order. Multi-User pricing is available.",
     -1, 13, 186, 204, 8
LTEXT "   Main Internet Address: cmcinter@suba.com",
     -1, 13, 203, 204, 8
LTEXT "   America Online: cmcinter@aol.com",
     -1, 13, 214, 204, 8
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 6, 6, 216, 228
ICON "MAIN_ICON", -1, 25, 27, 18, 20, SS_ICON | WS_GROUP
ICON "ORDER_HELP_ICON",-1,187,27,18,20, SS_ICON | WS_GROUP
CTEXT "PLEASE" include a LEGIBLE E-MAIL address with
all orders.",
     -1, 13, 98, 204, 8
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 16, 158, 198, 1
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 16, 93, 198, 1
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD |
     WS_VISIBLE, 16, 109, 198, 1
SEQUENCE DIALOG LOADONCALL MOVEABLE DISCARDABLE 27,37,237,147
STYLE DS_MODALFRAME | WS_POPUP| WS_CAPTION | WS_SYSMENU
CAPTION "Concatenate Files"
FONT 8, "MS Sans Serif"
EDITTEXT 750, 6, 16, 168, 12
DEFPUSHBUTTON "OK", 1, 183, 9, 45, 13
PUSHBUTTON "-> &Encode", 756, 183, 67, 45, 13
PUSHBUTTON "-> &Decode", 757, 183, 85, 45, 13
LISTBOX 751, 6, 44, 64, 82, LBS_STANDARD | WS_TABSTOP
PUSHBUTTON "->", 752, 77, 65, 18, 13
PUSHBUTTON "<-", 753, 77, 88, 18, 13
LISTBOX 754, 111, 44, 64, 82, LBS_STANDARD | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 183, 27, 45, 13
PUSHBUTTON "&Help", 755, 183, 45, 45, 13
LTEXT "Concatenate all files into:", -1, 6, 6, 87, 8
LTEXT "Files:", -1, 6, 33, 63, 8
LTEXT "Sequence:", -1, 111, 33, 63, 8
CTEXT "1", -1, 99, 45, 10, 8
CTEXT "2", -1, 99, 53, 10, 8
CTEXT "3", -1, 99, 61, 10, 8
CTEXT "4", -1, 99, 69, 10, 8
CTEXT "5", -1, 99, 77, 10, 8
```

```
CTEXT "6", -1, 99, 85, 10, 8
CTEXT "7", -1, 99, 93, 10, 8
CTEXT "8", -1, 99, 101, 10, 8
CTEXT "9", -1, 99, 109, 10, 8
RTEXT "...", -1, 99, 117, 10, 8
LTEXT "Status:", -1, 6, 132, 27, 8
LTEXT "", 758, 36, 132, 195, 8
PUSHBUTTON ">>", 759, 77, 45, 18, 13
PUSHBUTTON "<<", 760, 77, 109, 18, 13
LTEXT "Count:", -1, 183, 118, 24, 8
LTEXT "", 761, 210, 118, 21, 8
LTEXT "File", -1, 183, 109, 48, 8

WIN_CONFIG DIALOG LOADONCALL MOVEABLE DISCARDABLE
     25, 21, 267, 186
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "CMCCODE Options"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 213, 9, 45, 13
CHECKBOX "C&reate Report File", 401, 12, 9, 78, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "St&art as Icon", 402, 12, 23, 78, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "Auto File & Overwrite", 403, 12, 37, 78, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "A&lways On Top", 405, 93, 9, 75, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "Close When Do&ne", 406, 93, 23, 75, 12,
     BS-AUTOCHECKBOX | WS_TABSTOP
CHECKBOX "&Memory Swapping", 407, 93, 37, 75, 12
CHECKBOX "W&insort First", 408, 93, 51, 75, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
COMBOBOX 409, 117, 72, 54, 39, CBS_DROPDOWNLIST |
     WS_VSCROLL | WS_TABSTOP
COMBOBOX 410, 117, 87, 54, 39, CBS_DROPDOWNLIST |
     WS_VSCROLL | WS_TABSTOP
EDITTEXT 411, 12, 114, 141, 9, ES_AUTOHSCROLL |
     NOT WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 412, 156, 110, 12, 13
EDITTEXT 413, 15, 156, 148, 21, ES_MULTILINE |
     WS_BORDER | WS_VSCROLL | WS_TABSTOP
RADIOBUTTON "&Wincode Default", 414, 183, 94, 69, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "&Custom:", 415, 183, 109, 39, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 416, 224, 109, 30, 12
RADIOBUTTON "&Standard (Default)", 417, 183, 146, 75, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "Cus&tom:", 418, 183, 163, 39, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 419, 224, 163, 30, 12
CHECKBOX "DOS Attri&butes", 404, 12, 51, 78, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 213, 27, 45, 13
```



```
PUSHBUTTON "&Defaults", 420, 213, 45, 45, 13
PUSHBUTTON "&Help", 421, 213, 63, 45, 13
GROUPBOX "Working Directory", 103, 6, 102, 165, 24,
     BS_GROUPBOX
LTEXT "Enter sixty-four valid ASCII characters.",
     -1, 15, 145, 132, 9
CONTROL "", -1, "STATIC", SS_BLACKFRAME | WS_CHILD
     | WS_VISIBLE, 6, 6, 165, 60
GROUPBOX "Mode", 101, 177, 81, 84, 45, BS_GROUPBOX
GROUPBOX "Line Length", 102, 177, 130, 84, 51, BS_GROUPBOX
GROUPBOX "Code Table", -1, 6, 130, 165, 51, BS_GROUPBOX
LTEXT "Interactive Mode Setting:", -1, 6, 90, 90, 8
LTEXT "Sound Effects Setting:", -1, 6, 76, 81, 8
}
WNS_CONFIG DIALOG LOADONCALL MOVEABLE DISCARDABLE
     22, 38, 255, 159

STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Winsort Options"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 201, 9, 45, 13
EDITTEXT 501, 42, 19, 126, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
EDITTEXT 502, 42, 35, 126, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
CHECKBOX "&Use Custom BEGIN/END", 503, 12, 54, 123, 12,
     BS_AUTOCHECKBOX | WS_TABSTOP
EDITTEXT 504, 12, 93, 117, 9, ES_AUTOHSCROLL |
     NOT WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 505, 132, 89, 12, 13
EDITTEXT 506, 12, 121, 117, 9, ES_AUTOHSCROLL |
     NOT WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 507, 132, 117, 12, 13
CHECKBOX "Execute Winsort in Silent & Mode",
     508, 9, 140, 138, 12, BS_AUTOCHECKBOX | WS_TABSTOP
RADIOBUTTON "&Standard Winsort", 509, 159, 95,
     75, 12, BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "Flush &Left ONLY", 510, 159, 110,
     75, 12, BS_AUTORADIOBUTTON | WS_TABSTOP
RADIOBUTTON "Flush Left and Sort", 511, 159, 125,
     75, 12, BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 512, 216, 139, 24, 12
PUSHBUTTON "Cancel", 2, 201, 27, 45, 13
PUSHBUTTON "&Defaults", 513, 201, 45, 45, 13
PUSHBUTTON "&Help", 514, 201, 63, 45, 13
LTEXT "END:", -1, 12, 38, 24, 8
LTEXT "BEGIN:", -1, 12, 22, 27, 8
GROUPBOX "Sort Options", 101, 153, 81, 96, 72, BS_GROUPBOX
GROUPBOX "Custom BEGIN/END", -1, 6, 6, 168, 66, BS_GROUPBOX
GROUPBOX "Winsort Executable", 102, 6, 81, 141, 24, BS_GROUPBOX
GROUPBOX "Winsort Directory", 103, 6, 109, 141, 24, BS_GROUPBOX
LTEXT "Flush # Chars:", -1, 159, 141, 54, 8
}
Z_UZ_CONFIG DIALOG LOADONCALL MOVEABLE DISCARDABLE
```

```
     27, 24, 240, 151
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "ZIP/UNZIP Options"
FONT 8, "MS Sans Serif"
{
DEFPUSHBUTTON "OK", 1, 186, 9, 45, 13
EDITTEXT 601, 69, 6, 90, 12, ES_AUTOHSCROLL | WS_BORDER
     | WS_TABSTOP
PUSHBUTTON "?", 602, 162, 5, 12, 13
EDITTEXT 603, 69, 23, 105, 12, ES_AUTOHSCROLL | WS_BORDER
     | WS_TABSTOP
EDITTEXT 604, 69, 39, 90, 12, ES_AUTOHSCROLL | WS_BORDER
     | WS_TABSTOP

PUSHBUTTON "?", 605, 162, 38, 12, 13
EDITTEXT 606, 69, 55, 105, 12, ES_AUTOHSCROLL | WS_BORDER
     | WS_TABSTOP
EDITTEXT 607, 102, 71, 33, 12
RADIOBUTTON "Def&ault to location of input file", 608, 12,
     100, 132, 12, BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "User select &on UNZIP", 609, 12,
     114, 132, 12, BS_AUTORADIOBUTTON | WS_TABSTOP
RADIOBUTTON "&Set:", 610, 12, 128, 27, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
EDITTEXT 611, 42, 128, 99, 12, ES_AUTOHSCROLL |
     WS_BORDER | WS_TABSTOP
PUSHBUTTON "?", 612, 144, 128, 12, 13
RADIOBUTTON "&Normal", 613, 171, 100, 57, 12,
     BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP
RADIOBUTTON "&Minimized", 614, 171, 114, 57, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
RADIOBUTTON "Hidd&en", 615, 171, 128, 57, 12,
     BS_AUTORADIOBUTTON | WS_TABSTOP
PUSHBUTTON "Cancel", 2, 186, 27, 45, 13
PUSHBUTTON "&Defaults", 616, 186, 45, 45, 13
PUSHBUTTON "&Help", 617, 186, 63, 45, 13
GROUPBOX "UNZIPped File(s) Directory", 101, 6, 87, 153, 57,
     BS_GROUPBOX
GROUPBOX "Show Options", 102, 165, 87, 69, 57, BS_GROUPBOX
LTEXT "ZIP Filename:", -1, 6, 10, 57, 8
LTEXT "ZIP Param(s):", -1, 6, 26, 57, 8
LTEXT "UNZIP Filename:", -1, 6, 42, 60, 8
LTEXT "UNZIP Param(s):", -1, 6, 58, 60, 8
LTEXT "Extension for ZIPped Files:", -1, 6, 74, 93, 8
}
ZIP_NAME DIALOG LOADONCALL MOVEABLE DISCARDABLE 35, 31, 132, 60
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "CMCCODE - ZIP Filename"
FONT 8, "MS Sans Serif"
{
EDITTEXT 101, 31, 23, 51, 12, ES_AUTOHSCROLL | WS_BORDER
     | WS_TABSTOP
DEFPUSHBUTTON "OK", 1, 12, 42, 45, 13
PUSHBUTTON "Cancel", 2, 75, 42, 45, 13
CTEXT "Enter a filename for the ZIP archive:", -1, 4, 7, 123, 9
```



```
LTEXT "", 102, 84, 25, 24, 8
}
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
{
101, "Encode a data file..."
102, "Decode a data file..."
103, "Concatenate multiple files into a single file
     (specific ordering)..."

104, "View a Wincode Report file..."
105, "Clean Wincode directories by deleting files..."
106, "Set Encode options..."
107, "Set Decode options..."
108, "Set General Wincode options..."
109, "Set Winsort options..."
110, "Select a Report File viewer..."
111, "Set PKZIP/UNZIP options..."
}
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
{
112, "Set the Application Hook options..."
113, "Wincode Help Contents..."
114, "Help file Keyword Search..."
115, "Help on using Windows Help files..."
116, "Wincode Internet Frequency Asked Questions..."
117, "Legal Copyrights for files..."
118, "Information on ordering the Wincode Help file..."
119, "Version and Author information..."
121, "Set Wincode Interactive Drag & Drop Mode..."
122, "Hook the Wincode Menu into a selected application..."
123, "Select the Options Toolbar to configure Wincode..."
124, "Exit the Wincode program..."
125, "Stop the current Operation..."
126, "Quit the entire Operation..."
127, "Encode, Decode, Exit..."
}
STRINGTABLE LOADONCALL MOVEABLE DISCARDABLE
{
128, "Concat, View, Clean, Drag&Drop Mode, Hook..."
129, "Encode, Decode, Wincodt, Winsort, Viewer,
     PKZIP/UNZIP, Hook App..."
130, "Help and related information..."
}
CLEAN_DOWN ICON LOADONCALL MOVEABLE DISCARDABLE


          COPYRIGHT - 1996  PLANET GRAPHICS, INC.
```

The following is a second software listing for the catcher program of the invention.


```
//--------------------------------------------------------
//
//
//          File:          NPSHELL.CPP
//
//
//          Advanced Features:
//                              + Secured trigger/key-access processes
//                              + Uncrippling Media files
//
//          Copyright © 1996-1997 HyperLOCK Technologies, Inc.
//          All Rights Reserved.
//
//--------------------------------------------------------
//
//                              HyperCD
//
//          The architecture of HyperCD allows for
//          authorized and secure rendering of crippled multimedia files
//          from the fastest link.  Such media may reside on a HyperCD
//          CD-ROM/DVD-ROM, a server through regular phone line,
//          broad-band fiber optics or satelite for speedy access.
//          The HyperCD media files are protected by crippling the
//          media.  Only server authorized user can obtain
//          trigger/keys/missing pieces from the server to unlock the
//          HyperCD media.
//
//          See Patent Application for details.
//
//--------------------------------------------------------

#ifndef _WIN32
#define _WIN32
#endif

#ifndef _NPAPI_H_
#include "npapi.h"
#include "plgwnd.h"
#include "CHyperCD.h"
#endif

#include <mmsystem.h>
#include <qtw.h>
#include <time.h>
#include <string.h>
```

```
#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>

//-------------------------------------------------------------------
// NPP_Initialize:
//-------------------------------------------------------------------
NPError NPP_Initialize(void)
{
        DEBUG_TEST("NPP_Initialize")
   return NPERR_NO_ERROR;
}


//-------------------------------------------------------------------
// NPP_Shutdown:
//-------------------------------------------------------------------
void NPP_Shutdown(void)
{
        DEBUG_TEST("NPP_Shutdown")
        return;
}
//-------------------------------------------------------------------
// NPP_New:
//-------------------------------------------------------------------

NPError NP_LOADDS
NPP_New(NPMIMEType pluginType,
            NPP instance,
            uint16 mode,
            int16 argc,
            char* argn[],
            char* argv[],
            NPSavedData* saved)
{
        DEBUG_TEST("NPP_New")

   if (instance == NULL)
      return NPERR_INVALID_INSTANCE_ERROR;

   instance->pdata = NPN_MemAlloc(sizeof(PluginInstance));
   PluginInstance* This = (PluginInstance*) instance->pdata;
   if (This != NULL)
   {
      This->window = NULL;
```



```
      This->cHypercd = new CHyperCD();

      This->mode = mode;
            This->bAutoStart = FALSE;
      This->bLoop = FALSE;
      strcpy( This->InformationField,"HyperCD");

            int idx;
            STRING sSYSFILE;

            strcpy(sSYSFILE,SYSFILE);
            char *p1,*p2;
            STRING szArg, szValue,cd_title;

            for ( idx =0; idx<argc; idx++) {
                  strcpy(szArg,  argn[idx]);
                  strcpy(szValue, argv[idx]);

                  // Check web tags and set HyperCD flags
                  SetHyperCDFlags(szArg, szValue);

      if(bDemandHyperCD)
        goto parsing_embed_tags;


            for ( idx =0; idx<argc; idx++) {
                  strcpy(szArg,  argn[idx]);
                  strcpy(szValue, argv[idx]);

                  ParseHyperCDTags1(szArg,szValue);
                  }

            if(!bDemandHyperCD)
                  SysIO(sSYSFILE);

      for (idx =0; idx<argc; idx++)
        {
        strcpy(szArg,  argn[idx]);
                  strcpy(szValue, argv[idx]);

        ParseHyperCDTags2(szArg,szValue);
                  }
parsing_embed_tags:
                  instance->pdata = This;
                  return NPERR_NO_ERROR;
```

```
    }
  else
    return NPERR_OUT_OF_MEMORY_ERROR;
}



static void UnSubclass(PluginInstance *This)
{
  WNDPROC    OldWndProc;
  WNDPROC*   lplpfn = This->window->GetSuperWndProcAddr();
      DEBUG_TEST("UnSubclass")

  if ( !*lplpfn )
  {
    ASSERT(0);
    return;
  }

  // Set the original window procedure
  OldWndProc = (WNDPROC)::SetWindowLong( This->window->m_hWnd,
            GWL_WNDPROC, (LONG) *lplpfn );

  // A subclassed window's procedure is always AfxWndProc.
  // If this is not TRUE, then it's not a subclassed window.
  if ( OldWndProc != AfxWndProc )
    ASSERT(0);
}

static void KillHyperCDWindow(PluginInstance *This)
{

      DEBUG_TEST("KillHyperCDWindow")


  if (This->cHypercd) {
    This->cHypercd->Close();
    delete This->cHypercd;
    This->cHypercd = NULL;
      }

      CleanUpHyperCD();
  UnSubclass(This);
```



```
    if (This->window) {
    This->window->Detach();
    delete This->window;
    This->window = NULL;
  }
}



//----------------------------------------------------------------------
// NPP_Destroy:
//----------------------------------------------------------------------
NPError NP_LOADDS
NPP_Destroy(NPP instance, NPSavedData** save)
{
      if (instance == NULL)
    return NPERR_INVALID_INSTANCE_ERROR;

  PluginInstance* This = (PluginInstance*) instance->pdata;


      //
  // Note: If desired, call NP_MemAlloc to create a
  // NPSavedData structure containing any state information
  // that you want restored if this plugin instance is later
  // recreated.
  //


  if (This != NULL)
  {
          KillHyperCDWindow(This);
    NPN_MemFree(instance->pdata);
  }

  return NPERR_NO_ERROR;
}



//----------------------------------------------------------------------
// NPP_SetWindow:
//----------------------------------------------------------------------
NPError NP_LOADDS
NPP_SetWindow(NPP instance, NPWindow* np_window)
{
```

```
    DEBUG_TEST("NPP_SetWindow")


if (instance == NULL)
  return NPERR_INVALID_INSTANCE_ERROR;

PluginInstance* This = (PluginInstance*) instance->pdata;

//
// Note: Before setting fWindow to point to the
// new window, you may wish to compare the new window
// info to the previous window (if any) to note window
// size changes, etc.
//

if (!np_window)
  return NPERR_GENERIC_ERROR;

if (!instance)
  return  NPERR_INVALID_INSTANCE_ERROR;

if (!This)
  return NPERR_GENERIC_ERROR;

if (!np_window->window && !This->window) // spurious entry
  return NPERR_NO_ERROR;

if (!np_window->window && This->window)
{  // window went away
  KillHyperCDWindow(This);
  return NPERR_NO_ERROR;
}
if (!This->window && np_window->window)
{  // First time in -- no window created by plugin yet
  This->window = (CPluginWindow *) new CPluginWindow();
  if (!This->window->SubclassWindow((HWND)np_window->window))
  {
    MessageBox(NULL,"SubclassWindow Failed","HyperCD",MB_OK);
    return NPERR_GENERIC_ERROR;
  }
  // Save This pointer in window class member variable,.this lets the
  // window message handling have access to the data pointer easily
  This->window->StoreData(This);
```



```
  }

  // resize or moved window (or newly created)
  This->window->InvalidateRect(NULL);
  This->window->UpdateWindow();
  return NPERR_NO_ERROR;
}


//------------------------------------------------------------------
// NPP_NewStream:
//------------------------------------------------------------------
NPError NP_LOADDS
NPP_NewStream(NPP instance,
              NPMIMEType type,
              NPStream *stream,
              NPBool seekable,
              uint16 *stype)
{

    DEBUG_TEST("NPP_NewStream")


  if (instance == NULL)
    return NPERR_INVALID_INSTANCE_ERROR;
  PluginInstance* This = (PluginInstance*) instance->pdata;

  *stype = NP_ASFILE;

  return NPERR_NO_ERROR;
}


int32 STREAMBUFSIZE = 0X0FFFFFFF;  // If we are reading from a file in
NPAsFile
                  // mode so we can take any size stream in our
                  // write call (since we ignore it)

//------------------------------------------------------------------
// NPP_WriteReady:
//------------------------------------------------------------------
int32 NP_LOADDS
NPP_WriteReady(NPP instance, NPStream *stream)
```

```
{
        DEBUG_TEST("NPP_WriteReady")

   if (instance != NULL)
   {
      PluginInstance* This = (PluginInstance*) instance->pdata;
   }
   return STREAMBUFSIZE;  // Number of bytes ready to accept in
NPP_Write()
}



//--------------------------------------------------------
// NPP_Write:
//--------------------------------------------------------
int32 NP_LOADDS
NPP_Write(NPP instance, NPStream *stream, int32 offset, int32 len, void *buffer)
{
        DEBUG_TEST("NPP_Write")


   if (instance != NULL)
   {
      PluginInstance* This = (PluginInstance*) instance->pdata;

   }

   return len;          // The number of bytes accepted
}
//--------------------------------------------------------
// NPP_DestroyStream:
//--------------------------------------------------------
NPError NP_LOADDS
NPP_DestroyStream(NPP instance, NPStream *stream, NPError reason)
{
        DEBUG_TEST("NPP_DestroyStream")

   if (instance == NULL)
      return NPERR_INVALID_INSTANCE_ERROR;
   PluginInstance* This = (PluginInstance*) instance->pdata;
```



```
   return NPERR_NO_ERROR;
}

//--------------------------------------------------------
//
//      HCD_To_Server
//
//      This module is called by various components of HyperCD client software
//      to initiate communications with server(s).
//      Objects will be exchanged during such process.
//
//--------------------------------------------------------
HCDError HCD_To_Server(HCDOBJECTTYPE InObject, HCDOBJECTTYPE
OutObject, HCDCOMMTYPE CommType )
{
        HCDError  HCDReturnErr=HCDOK;

        // retrieve objects from server and assigned to OutObject
        // this client is identified by InObject
        // NPN_ methods can be used.
        if(CommType == HCD_GET)
                HCDReturnErr=HCD_GetURL(InObject,OutObject);

        // Send info to server
        else if (CommType == HCD_POST)
                HCDReturnErr=HCD_PostURL(InObject,OutObject);

        else
                HCDReturnErr=HCD_DefaultComm(InObject,OutObject);

        return HCDReturnErr;

}
//--------------------------------------------------------
// ObtainKey:
//--------------------------------------------------------

HCDError ObtainKey(KeyObject *Object,  char *pszTrigger)
{
        TriggerObject Trigger;


        //*************************************************************
***
        //
```

```
        //      Decrypt/decode trigger to obtain info on
        //      server id, communication keys,
        //      time-stamp, etc that make the communication
        //      secure and unique so that intercepted
        //      keys will not work on PIRATED SERVER
        //      but fees/usage info can still be charged/monitored
        //      on PIRATED HyperCD's.
        //

        //*************************************************************
***

        Trigger = DecryptTrigger(pszTrigger);
        if(IsValidTrigger(TestTrigger))
                return HCDERROR_UNAUTHORIZED_ACCESS;


        //*************************************************************
***
        //
        //      Directly communicate with the server
        //      to obtain CRITICAL DATA - the server approved missing
        //      pieces.
        //
        //      The CRITICAL DATA includes, but not limited to,
        //      the HEADER, jumptable, parts of the data.
        //      The HEADER here includes organization
        //      information which specifies where the sub data chunks are,
        //      what leading keywords are, what encryption is
        //      performed on sub data chunks,  and what access level
        //      numbers are derived from a formula used to characterize
        //      a set of data.
        //
        //      This component is also called the Catcher that
        //      captures the CRITICAL DATA.
        //

        //*************************************************************
***

        NewObject = new KeyObject;

        // Initiate a process to retrieve the missing uncrippling
        //      pieces to merge with the crippled media file on
        //      HyperCD/HyperCD media server/any other convenient
```



```
        //      locations where speedy delivery is possible.

        RetrieveKeys(Trigger, NewObject);

        if(IsValidKeyObject(NewObject))
                return HCDERROR_UNAUTHORIZED_ACCESS;
        return HCDOK;
}


//-----------------------------------------------------------
// NPP_StreamAsFile:
//-----------------------------------------------------------
void NP_LOADDS
NPP_StreamAsFile(NPP instance, NPStream *stream, const char* pszTrigger)
{

        if (instance == NULL)
                return;
        PluginInstance* This = (PluginInstance*) instance->pdata;
        if (!This->cHypercd)
                return;


        //*************************************************************
***
        //
        //      Direct Access to obtain missing map/object/keys
        //
        //      This component decrypts the incoming trigger
        //      and then perform Secured Access by directly
        //      communicates with the server to obtain
        //      the missing critical information
        //      The crippled file is then being uncrippled
        //      and then sent to media display component
        //      for rendering.
        //
        //      The incoming keys may be from several servers.
        //
        //      The incoming keys may be of type:
        //      * HyperCD Triggers indicating embedded HyperCD Objects
        //      * Server  Keys facilitating secure communication
        //      * Server  Keys dictating what objects to expect and
        //                      what opeartions to perform on them
        //      * Server  Keys that's uniquely protected/encrypted
```

```
//                  to deliver missing uncrippling
//                  parts/objects
//

//*************************************************************
***

ObtainKey(pszTrigger, Key, KeyType);
if(KeyType == HyperCDKey_EmbedTrigger)
        {
        PrepareEmbedHyperCD(Key);
        return;
        }

//*************************************************************
***
//
//      After the server receives a POST request from
//      end-user client software - asking the
//      permission to access HyperCD media on end-user
//      computer,  the server checks for
//      registration/access permission info on the user
//      and then setup a Secure Communication channel
//      with the end-user client software
//
//*************************************************************
***

else if (KeyType == HyperCDKey_SecureComm)
        {
        // Save server-ID, time-stamp, password info
        SetupSecureComm(Key);
        return;
        }
else if (KeyType == HyperCDKey_ActionPlan)
        {
        // Save info on objects/operations/jump table/etc
        SetupActionPlan(Key);
        return;
        }
else if (KeyType == HyperCDKey_Objects)
        {
        // Receive objects
        ReceiveObjects(Key,Object);
```



```
//*************************************************************
***
//
//      Media Display Component
//
//      This component uncripples the crippled files from
//      HyperCD and display them.  The uncrippling is achieved
//      by decrypting/remapping/reorganization of the parts from
//      remote server and HyperCD, which could reside on a DVD
//      a server, or any media type.
//

//*************************************************************
***

        // Check if the intended file is of HCD_MEDIA
        if( ObjectType(Object) == HCD_MEDIA) // mov, jpeg, avi, and
other encrypted media type
                {
                // Uncrippling media files from HyperCD and render the
files
                DisplayObject(Object );
                }
        else // anything else is not valid
                {
                HyperCDError(INVALID_MEDIA_TYPE);
                return;
                }
        return;
        }
    else    // for keys of other types,  perform default processing
        {
        PerformDefaultProcessing();
        return;
        }
    return;
}


//-------------------------------------------------------------
// NPP_Print:
//-------------------------------------------------------------
void NP_LOADDS
```

```
NPP_Print(NPP instance, NPPrint* printInfo)
{
        DEBUG_TEST("NPP_Print")

  if(printInfo == NULL)  // trap invalid parm
    return;

  if (instance != NULL)
  {
    PluginInstance* This = (PluginInstance*) instance->pdata;

    if (printInfo->mode == NP_FULL)
    {
      //
      // Note: If your plugin would like to take over
      // printing completely when it is in full-screen mode,
      // set printInfo->pluginPrinted to TRUE and print your
      // plugin as you see fit.  If your plugin wants Netscape
      // to handle printing in this case, set printInfo->pluginPrinted
      // to FALSE (the default) and do nothing.  If you do want
      // to handle printing yourself, printOne is true if the
      // print button (as opposed to the print menu) was clicked.
      // On the Macintosh, platformPrint is a THPrint; on Windows,
      // platformPrint is a structure (defined in npapi.h) containing
      // the printer name, port, etc.
      //
      void* platformPrint = printInfo->print.fullPrint.platformPrint;
      NPBool printOne = printInfo->print.fullPrint.printOne;

      printInfo->print.fullPrint.pluginPrinted = FALSE; // Do the default

    }
    else    // If not fullscreen, we must be embedded
    {
      //
      // Note: If your plugin is embedded, or is full-screen
      // but you returned false in pluginPrinted above, NPP_Print
      // will be called with mode == NP_EMBED.  The NPWindow
      // in the printInfo gives the location and dimensions of
      // the embedded plugin on the printed page.  On the Macintosh,
      // platformPrint is the printer port; on Windows, platformPrint
      // is the handle to the printing device context.
      //
      NPWindow* printWindow = &(printInfo->print.embedPrint.window);
      void* platformPrint = printInfo->print.embedPrint.platformPrint;
```



```
    }
  }
}


//------------------------------------------------------------------------
// NPP_HandleEvent:
// Mac-only.
//------------------------------------------------------------------------
int16 NPP_HandleEvent(NPP instance, void* event)
{
  NPBool eventHandled = FALSE;
  if (instance == NULL)
    return eventHandled;

  PluginInstance* This = (PluginInstance*) instance->pdata;

  //
  // Note: The "event" passed in is a Macintosh
  // EventRecord*.  The event.what field can be any of the
  // normal Mac event types, or one of the following additional
  // types defined in npapi.h: getFocusEvent, loseFocusEvent,
  // adjustCursorEvent.  The focus events inform your plugin
  // that it will become, or is no longer, the recepient of
  // key events.  If your plugin doesn't want to receive key
  // events, return false when passed at getFocusEvent.  The
  // adjustCursorEvent is passed repeatedly when the mouse is
  // over your plugin; if your plugin doesn't want to set the
  // cursor, return false.  Handle the standard Mac events as
  // normal.  The return value for all standard events is currently
  // ignored except for the key event: for key events, only return
  // true if your plugin has handled that particular key event.
  //

  return eventHandled;
}
```

```
//*************************************************************
//
//              HyperCD I/O
//
//    Function:
//              Remap-decrypt-decode and merge the
//              missing CRITICAL DATA with the CRIPPLED
//              HyperCD files.
//
//*************************************************************

#include <windows.h>
#include <mmsystem.h>
#include <digitalv.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <string.h>

InsertHyperCDIOModule();
RemoveHyperCDIOModule();
OPEN_HYPERCD();

HDVDCALLBACK HYPERCDIO(LPHYPERCDINFO)
{
  switch (uMessage) {
        case MMIOM_OPEN:
              HYPERCD_Open();
              return 0;

        case MMIOM_CLOSE:
              HYPERCD_Close();
              return 0;

        case MMIOM_READ:
              HYPERCD_ReadMultipleFiles();
              return (LPHYPERCDINFO.displacement);

        case MMIOM_SEEK:
              switch (lParam2) {
                case SEEK_SET:   // seek to the absolute position relative to
original beginning
                      HYPERCD_SeekSet();
                      break;
```



```
/*-------------------------------------------------------

     File:          cHyperCD.cpp

     Advanced Features:

          This file implements a CHyperCD class which can be
          used to display HyperCD movie files.  This file
          contains some basic code from the Netscape plugin
          sdk.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#include "stdafx.h"
#include "CHyperCD.h"
#include <mmsystem.h>

#ifdef WIN32
    #include <digitalv.h>
endif

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[ ] = _FILE_;
endif


* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        The constructor
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

HyperCD: :CHyperCD ( )

     mOpen = FALSE;
     mPlaying = FALSE;
     mDeviceID = 0;
     mErrorCode = 0;
     mMCIErrorCode = 0L;
}

CHyperCd::~CHYperCD ( )
{
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
This function opens the HyperCD movie file for playback and
display the first frame.  It requires the HyperCD movie file
name and a pointer to the window to draw into
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

BOOL CHyperCD: :Open (CWnd *pWnd, CString Filename, CString Type)
{
     DWORD RetCode;
     MCI_ANIM_OPEN_PARMS OpenParms;
     MCI_ANIM_WINDOW_PARMS WindowParms;
     // Close any device that is already open.
     if (mDeviceID) {
```

```
        Close ( );
        }

    // Open a device for playback.
    OpenParams.dwCallback = NULL;
    OpenParams.wDeviceID = 0
    OpenParams.lpstrDeviceType = Type;
    OpenParams.lpstrAlias = "hyperCD";// can use an array
         of aliases
    OpenParams.dwStyle = WS_CHILD | WS_VISIBLE;
    OpenParams.hWndParent = pWnd->m_hWnd;
    DWORD flags=(DWORD) MCI_OPEN_ELEMENT |
    MCI_ANIM_OPEN_PARENT | MCI_ANIM_OPEN_WS;
    if (Type =="AVIVideo" ||Type=="QTWVideo"||Type=="MPEGVideo")
         flags |= MCI_OPEN_TYPE;
    if (RetCode - mciSendCommand (0,MCI_OPEN, flags, (DWORD)
    (LPVOID) &OpenParms) ) {
         mMCIErrorCode = RetCode;
         char szBuf [256]
         mciGetErrorString(Ret Code.szBuf,256);
         char msg[200];
         strcpy(msg,  "Error Opening : "); strcat(msg.
         Filename):
         MessageBox(NULL.szBuf.msg.MB_OK);
         return FALSE;
         }

    // The device was opened, get the device ID.
    mDeviceID = OpenParms.wDeviceID;
    WindowParms.dwCallback = NULL;
    WindowParms.hWnd = pWnd->m_hWnd;
    WindowParms.nCmdShow = SW_SHOW;
    WindowParms.lpstrText = (LPSTR) NULL;
    if (RetCode = mciSendCommand (mDeviceID, MCI_WINDOW,
    MCI_ANIM_WINDOW-HWND, (DWORD) (LPVOID) & WindowParms)) {

         mMCIErrorCode = RetCode;
         return FALSE;
    }
    mMovieWnd = WindowParms.hWnd;

    mOpen = TRUE
    return TRUE;
}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    Stop any HyperCD movie, close any open device IDs.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void CHyperCD: :Close (void)
{
    // Closing a device ID will stop the video playback.
    if (mDeviceID)
         mciSendCommand (mDeviceID, MCI_CLOSE, 0L, NULL):

    mOpen = FALSE
    mPlaying = FALSE;
```



```
}
void CHyperCD: :Update ( )
{
    MCI_ANIM_UPDATE_PARMS UpdateParams;

    UpdateParams.dwCallback = NULL
    UpdateParams.hDC = ::GetDC(mMovieWnd);
    if (mDevice ID)
    mciSendCommand (mDeviceID, MCI_UPDATE,
    MCI_ANIM_UPDATE_HDC, (DWORD) (LPVOID)&UpdateParams);

    ::ReleaseDC (mMovieWnd,UpdateParams.hDC);
    mOpen = FALSE;
    mPlaying = FALSE;
}
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    Start the video playback.
    This function immediately returns control back to the
    program.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

BOOL CHyperCD::Start (BOOL bLoop)
{
    DWORD RetCode, dwFlags = 0L;
    MCI_ANIM_PLAY_PARMS  PlayParms;

    //Start playback using the MCI_PLAY command.
    PlayParms.dwCallback = NULL;
    PlayParms.dwFrom = PlayParms.dwTo = 0;

#ifdep WIN32
    if (bLoop) dwFlags = MCI_DGV_PLA_REPEAT;
#endif
    if (RetCode = mciSendCommand (mDeviceID, MCI_PLAY, dwFlags,
    (DWORD) (LPVOID) &PlayParms)) {
         mMCIErrorCode = RetCode;
         char szBuf[256];
         mciGetErrorString(RetCode,szBuf,256);
         mciSendCommand (mDeviceID, MCI_CLOSE, 0L, NULL);
         mOpen = FALSE;
         }
    mPlaying = TRUE;
    return TRUE;

    BOOL CHyperCD::Realize (void)

         //plugins must realize their palette as a
         background palette
         DWORD RetCode =
              mciSendCommand (mDeviceID, MCI_REALIZE,
              MCI_ANIM_REALIZE_BKGD, NULL);
         return RetCode;
}
```

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     Pause a video, different from close.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

BOOL CHyperCD::Stop (void)
{
     DWORD RetCode;

     // Stop playback by sending the MCI_PAUSE command.
     if (RetCode = mciSend Command (mDeviceID, MCI_PAUSE,
     OL, NULL)){
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, OL, NULL)){
          mOpen = FALSE;
          return FALSE;
          }
     mPlaying = FALSE;
     return TRUE;
}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     Rewind the video to the beginning and display the
     first fram.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

BOOL CHyperCD::Rewind (void)
{
     DWORD RetCode;

     // If the video is playing you must stop it first,
     if (mPlaying)
          if (!Stop( ))
               return FALSE;

     // Use the MCI_SEEK command to return to the beginning
     of the file.
     if (RetCode = mciSendCommand (mDeviceID, MCI_SEEK,
     MCI_SEEK_TO_START, (DWORD) (LPVOID) (NULL)){
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, OL, NULL);
          mOpen = FALSE;
          return = FALSE;
          }
     return TRUE;
}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     Forward the video to the end and display the last frame.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - *

BOOL CHyperCD::Forward (void)

     DWORD RetCode;

     // If the video is playing you must stop it first.
     if (mPlaying)
          if (!Stop ( ))
```



```
               return FALSE;

     // Use the MCI_SEEK command to go to the end of the file.
     If (RetCode = mciSendCommand (mDeviceID, MCI_SEEK,
     MCI_SEEK_TO_END, (DWORD) (LPVOID) NULL)){
     mMCIError Code = RetCode;
     mciSendCommand (mDeviceID, MCI_CLOSE, OL, NULL);
     mOpen = FALSE;
     return FALSE;
     }

     return TRUE;
}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     Forward the video by one frame.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

BOOL CHyperCD::FrameForward (void)
{
     DWORD RetCode;
     MCI_ANIM_STEP_PARMS StepParms;
     MCI_STATUS_PARMS StatusParms;
     DWORD Length, Position;

     // if the video is playing you must stop it first.
     if (mPlaying)
          if (!Stop ( ))
               return FALSE;

     // Determine the length in frames of the file.
     StatusParms.dwItem = MCI_STATUS_LENGTH;
     if (RetCode = mciSendCommand (mDeviceID, MCI_STATUS,
          MCI_STATUS_ITEM, (DWORD) (LPVOID) &StatusParms))
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, OL, NULL);
          return FALSE;
          }
     Length = StatusParms.dwReturn;

     //Determine the current position of the file.
     StatusParms.dwItem = MCI_STATUS_POSITION;
     if (RetCode = mciSendCommand (mDeviceID, MCI_STATUS,
     MCI_STATUS_ITEM, (DWORD) (LPVOID) &StatusParms)
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, OL, NULL);
          return FALSE;
          }
     Position = StatusParms.dwReturn;

     // If we're already at the end return.
     if (Length == Position)
          return TRUE;

     // If not already at the end use MCI_STEP to move
     forward one frame.
```

```
     StepParms.dwFrames = 1L;
     if (RetCode = mciSendCommand (mDeviceID, MCI_STEP,
     MCI_ANIM_STEP_FRAMES, (DWORD) (LPVOID) &StepParms))
{
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, 0L, NULL);
          mOpen = FALSE;
          return FALSE:
          }
     return TRUE;
}
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         Step back the video by one frame.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
BOOL CHyperCD::FrameBack (void)

     DWORD RetCode;
     MCI_ANIM_STEP_PARMS StepParms

     // If the video is playing you must stop it first.
     if (mPlaying)
          if (!Stop ( )0
               return FALSE;

     // Use MCI_STEP to move back one frame.
     StepParms.dwFrames = 1L;
     if (RetCode = mciSendCommand (mDeviceId, MCI_STEP,
     MCI_ANIM_STEP_REVERSE, (DWORD) (LPVOID) & StepParms))
{
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, 0L, NULL);
          mOpen = FALSE;
          return FALSE:
          }

     return TRUE;

DWORD CHyperCD::GetLength (void)

     DWORD RetCode;
     // Make sure a device is open.
     if (!mDeviceID)
          return 0;

     MCI_STATUS_PARMS StatusParms;

     //Determine the length in frames of the file.
     StatusParms.dwItem = MCI_STATUS_LENGTH;
     if (RetCode = mciSendCommand (mDeviceId, MCI-STATUS,
     MCI_STATUS-ITEM, (DWORD) (LPVOID) &StatusParms))
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, 0L, NULL);
          return FALSE;
          }
     return (int) StatusParms.dwReturn;
```



```
}
DWORD CHyperCD::GetPosition (void)
{
     DWORD RetCode;
     // Make sure a device is open.
     if (!mDeviceID)
          return 0;

     MCI_STATUS_PARMS StatusParms;

     // Determine the current position of the file.
     StatusParms.dwItem = MCI_STATUS_POSITION;
     if (RetCode = mciSendCommand (mDeviceID, MCI_STATUS,
     MCI_STATUS_ITEM, (DWORD) (LPVOID) &StatusParms))
          mMCIErrorCode = RetCode;
          mciSendCommand (mDeviceID, MCI_CLOSE, 0L, NULL);
          return FALSE;
          }
     return (int) StatusParms.dwReturn;
}

int CHyperCD::GetWidth (void)
{
     //Make sure a device is open.
     if (!mDeviceID)
          return 0;
     MCI_ANIM_RECT_PARMS RectParms;

     //Use MCI_WHERE to get the video window rectangle.
     mciSendCommand (mDeviceID, MCI_WHERE, (DWORD)
     MCI_ANIM_WHERE_SOURCE. (DWORD) (LPVOID) &RectParms);

     return (int) RectParms.rc.right;

int CHyperCD::GetHeight (void)

     // Make sure a device is open.
     if (!mDeviceID)
          return 0;

     MCI_ANIM_RECT_PARMS RectParms;

     // Use MCI_WHERE to get the video window rectangle.
     mciSendCommand (mDeviceId, MCI_WHERE, (DWORD)
     MCI_ANIM_WHERE_SOURCE, (DWORD) (LPVOID) &RectParms);

     return (int) RectParms.rc.bottom;

CString CHyperCD::GetError String (void)
{
     static const char "Strings[ ] = {
          "Could not set the position for the video
          in the window.".
          };
     char Error Buffer (MAXERRORLENGTH];
```

```
     // An error was generated from within the CHyperCD class.
     if (mErrorCode == 1)
          return (CString) Strings[0];
     // An error was generated from a MCI function call.
     else if (mciGetErrorString (mMCIErrorCode,  (LPSTR)
     ErrorBuffer,
     MAXERRORLENGTH))
          return (CString) Error Buffer;
     // There is no error.
     else.
          return (CString) ("There is no error or the error
          is undefined.")
}
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
A private function that simply positions the video window in
the center of the parent window.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
BOOL CHyperCD::Center (void)
{
     DWORD RetCode;
     CRect BoundsRect, MovieRect, WindowRect;
     MCI_ANIM_RECT_PARMS RectParms;

     // Use MCI_WHERE to get the video window rectangle.
     if (RetCode = mciSendCommand (mDeviceID, MCI_WHERE,
     (DWORD)
     MCI_ANIM_WHERE_SOURCE, (DWORD) (LPVOID) &RectParms))
          return FALSE;

     // Determine the parameters for the playback window.
     BoundsRect = RectParms.rc;
     MovieRect.left = 0;
     MovieRect.top = 0;
     MovieRect.right = MovieRect.left + BoundsRect.right;
     MovieRect.bottom = MovieRect.top + BoundsRect.bottom;

     ::GetWindowRect (mMovieWnd.&WindowRect);

     // Move the playback window.
     MoveWindow (mMovieWnd, (WindowRect.Width( ) -
     MovieRect.Width( ))/2,
                            (WindowRect.Height( ) -
                           MovieRect.Height)( ))/2
                           BoundsRect.right.
                           BoundsRect.bottom, TRUE);
     return TRUE;

}
```


The following is the software code listing for the requesting, end-user's computer for the embodiment of Fig. 12.

```
/*-------------------------------------------------------
//
//                          HyperKey
//
// The architecture of HyperKey allows for
// authorized and secure rendering of encrypted multimedia object
// from the protected web site.  The encrypted HyperKey media object is
protected
// by crippling the media.   Only authorized user can obtain
// trigger/keys from the server to unlock the HyperKey media.
//
-------------------------------------------------------*/


#ifndef _WIN32
#define _WIN32
#endif

#ifndef _NPAPI_H_
#include "npapi.h"
#include "plgwnd.h"
#include "CHyperCD.h"
#endif

#include <mmsystem.h>
#include <qtw.h>


#include <time.h>
#include <string.h>
#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>

//----------------------------------------------------------
// NPP_Initialize:
//----------------------------------------------------------
NPError NPP_Initialize(void)
{
        DEBUG_TEST("NPP_Initialize")
   return NPERR_NO_ERROR;
}
```



```
//----------------------------------------------------------
// NPP_Shutdown:
//----------------------------------------------------------
void NPP_Shutdown(void)
{
        DEBUG_TEST("NPP_Shutdown")
        return;
}
//----------------------------------------------------------
// NPP_New:
//----------------------------------------------------------

NPError NP_LOADDS
NPP_New(NPMIMEType pluginType,
            NPP instance,
            uint16 mode,
            int16 argc,
            char* argn[],
            char* argv[],
            NPSavedData* saved)
{
        DEBUG_TEST("NPP_New")

  if (instance == NULL)
    return NPERR_INVALID_INSTANCE_ERROR;

  instance->pdata = NPN_MemAlloc(sizeof(PluginInstance));
  PluginInstance* This = (PluginInstance*) instance->pdata;
  if (This != NULL)
  {
    This->window = NULL;
    This->cHypercd = new CHyperCD();

    This->mode = mode;
      for ( idx =0; idx<argc; idx++) {
                    strcpy(szArg,  argn[idx]);
                    strcpy(szValue, argv[idx]);

                    // Check web tags and set HyperKey flags
                    SetHyperKeyFlags(szArg, szValue);

    if(bDemandHyperKey)
      goto parsing_embed_tags;
```

```
            for ( idx =0; idx<argc; idx++) {
                    strcpy(szArg,  argn[idx]);
                    strcpy(szValue, argv[idx]);

                    ParseHyperKeyTags1(szArg,szValue);
                    }

            if(!bDemandHyperKey)
                    SysIO(sSYSFILE);

    for (idx =0; idx<argc; idx++)
      {
      strcpy(szArg,  argn[idx]);
            strcpy(szValue, argv[idx]);

      ParseHyperKeyTags2(szArg,szValue);
            }
parsing_embed_tags:
            instance->pdata = This;
            return NPERR_NO_ERROR;
      }
  else
    return NPERR_OUT_OF_MEMORY_ERROR;
}



static void UnSubclass(PluginInstance *This)
{
  WNDPROC    OldWndProc;
  WNDPROC*   lplpfn = This->window->GetSuperWndProcAddr();
      DEBUG_TEST("UnSubclass")

  if ( !*lplpfn )
  {
    ASSERT(0);
    return;
  }

  // Set the original window procedure
  OldWndProc = (WNDPROC)::SetWindowLong( This->window->m_hWnd,

      GWL_WNDPROC, (LONG) *lplpfn );
```



```
  // A subclassed window's procedure is always AfxWndProc.
  // If this is not TRUE, then it's not a subclassed window.
  if ( OldWndProc != AfxWndProc )
    ASSERT(0);
}

static void KillHyperCDWindow(PluginInstance *This)
{

      DEBUG_TEST("KillHyperCDWindow")


  if (This->cHypercd) {
    This->cHypercd->Close();
    delete This->cHypercd;
    This->cHypercd = NULL;
      }

      CleanUpHyperKey();
  UnSubclass(This);

      if (This->window) {
    This->window->Detach();
    delete This->window;
    This->window = NULL;
  }
}



//--------------------------------------------------------------------
// NPP_Destroy:
//--------------------------------------------------------------------
NPError NP_LOADDS
NPP_Destroy(NPP instance, NPSavedData** save)
{
      if (instance == NULL)
    return NPERR_INVALID_INSTANCE_ERROR;

  PluginInstance* This = (PluginInstance*) instance->pdata;


      //
  // Note: If desired, call NP_MemAlloc to create a
  // NPSavedData structure containing any state information
```

```
    // that you want restored if this plugin instance is later
    // recreated.
    //


    if (This != NULL)
    {
                KillHyperCDWindow(This);
       NPN_MemFree(instance->pdata);
    }

    return NPERR_NO_ERROR;
}


//--------------------------------------------------------------------
// NPP_SetWindow:
//--------------------------------------------------------------------
NPError NP_LOADDS
NPP_SetWindow(NPP instance, NPWindow* np_window)
{

        DEBUG_TEST("NPP_SetWindow")



    if (instance == NULL)
      return NPERR_INVALID_INSTANCE_ERROR;

    PluginInstance* This = (PluginInstance*) instance->pdata;

    //
    // Note: Before setting fWindow to point to the
    // new window, you may wish to compare the new window
    // info to the previous window (if any) to note window
    // size changes, etc.
    //

   if (!np_window)
       return NPERR_GENERIC_ERROR;

    if (!instance)
       return  NPERR_INVALID_INSTANCE_ERROR;

    if (!This)
```



```
       return NPERR_GENERIC_ERROR;

    if (!np_window->window && !This->window) // spurious entry
       return NPERR_NO_ERROR;

    if (!np_window->window && This->window)
    {  // window went away
       KillHyperCDWindow(This);
       return NPERR_NO_ERROR;
    }
    if (!This->window && np_window->window)
    {  // First time in -- no window created by plugin yet
       This->window = (CPluginWindow *) new CPluginWindow();
       if (!This->window->SubclassWindow((HWND)np_window->window))
       {
          MessageBox(NULL,"SubclassWindow Failed","HyperCD",MB_OK);
          return NPERR_GENERIC_ERROR;
       }
       // Save This pointer in window class member variable..this lets the
       // window message handling have access to the data pointer easily
       This->window->StoreData(This);
    }

    // resize or moved window (or newly created)
    This->window->InvalidateRect(NULL);
    This->window->UpdateWindow();
    return NPERR_NO_ERROR;
}


//--------------------------------------------------------------------
// NPP_NewStream:
//--------------------------------------------------------------------
NPError NP_LOADDS
NPP_NewStream(NPP instance,
                NPMIMEType type,
                NPStream *stream,
                NPBool seekable,
                uint16 *stype)
{

       DEBUG_TEST("NPP_NewStream")
```

```
  if (instance == NULL)
    return NPERR_INVALID_INSTANCE_ERROR;
  PluginInstance* This = (PluginInstance*) instance->pdata;

  *stype = NP_ASFILE;

  return NPERR_NO_ERROR;
}



int32 STREAMBUFSIZE = 0X0FFFFFFF;  // If we are reading from a file in
NPAsFile
                    // mode so we can take any size stream in our
                    // write call (since we ignore it)

//------------------------------------------------------------------
// NPP_WriteReady:
//------------------------------------------------------------------
int32 NP_LOADDS
NPP_WriteReady(NPP instance, NPStream *stream)
{

      DEBUG_TEST("NPP_WriteReady")

  if (instance != NULL)
  {
    PluginInstance* This = (PluginInstance*) instance->pdata;
  }
  return STREAMBUFSIZE;  // Number of bytes ready to accept in
NPP_Write()
}



//------------------------------------------------------------------
// NPP_Write:
//------------------------------------------------------------------
int32 NP_LOADDS
NPP_Write(NPP instance, NPStream *stream, int32 offset, int32 len, void *buffer)
{

      DEBUG_TEST("NPP_Write")
```



```
  if (instance != NULL)
  {
    PluginInstance* This = (PluginInstance*) instance->pdata;

  }

  return len;          // The number of bytes accepted
}
//------------------------------------------------------------------
// NPP_DestroyStream:
//------------------------------------------------------------------
NPError NP_LOADDS
NPP_DestroyStream(NPP instance, NPStream *stream, NPError reason)
{

      DEBUG_TEST("NPP_DestroyStream")

  if (instance == NULL)
    return NPERR_INVALID_INSTANCE_ERROR;
  PluginInstance* This = (PluginInstance*) instance->pdata;

  return NPERR_NO_ERROR;
}




//------------------------------------------------------------------
// NPP_StreamAsFile:
//------------------------------------------------------------------
void NP_LOADDS
NPP_StreamAsFile(NPP instance, NPStream *stream, const char* szStream)
{

      DEBUG_TEST("NPP_StreamAsFile")
      if (instance == NULL)
            return;
  PluginInstance* This = (PluginInstance*) instance->pdata;
  if (!This->cHypercd)
            return;

  // Obtain object/keys
      ObtainKey(Object,  szStream);
      if(bDemandHyperKey)
```

```
            {
            if(ObjectType(Object) == PGI_HYPERKEY) // file in
bDemandHyperKey

                  {// 1. check we have permission
                  Permission = PermissionFromWebtoUseHyperKey();

                  // if our right is lower than the permitted, return
                  if(Permission.right > User.right)
                        return;

                  // 2. if we have permission, get the access path
               strcpy(szPath,   GetAccessPath(Permission));

                  // 3. retrieve the object          and wait
                  RetrieveHyperKeyObject(szPath);
         return;
         }

            // B. check if the streamed file is PGI_MEDIA
            else if( ObjectType(Object) == PGI_MEDIA) // jpeg, avi,
encrypted media type
                  {
                  DisplayObject(Object  );
                  }
            else // anything else under bDemandHyperKey, is not valid
                  return;

            }
      else// for this version, return and not process other command
            return;

}



//-------------------------------------------------------------------
// NPP_Print:
//-------------------------------------------------------------------
void NP_LOADDS
NPP_Print(NPP instance, NPPrint* printInfo)
{
      DEBUG_TEST("NPP_Print")

  if(printInfo == NULL)   // trap invalid parm
```



```
      return;

   if (instance != NULL)
   {
      PluginInstance* This = (PluginInstance*) instance->pdata;

      if (printInfo->mode == NP_FULL)
      {
         //
         // Note: If your plugin would like to take over
         // printing completely when it is in full-screen mode,
         // set printInfo->pluginPrinted to TRUE and print your
         // plugin as you see fit.  If your plugin wants Netscape
         // to handle printing in this case, set printInfo->pluginPrinted
         // to FALSE (the default) and do nothing.  If you do want
         // to handle printing yourself, printOne is true if the
         // print button (as opposed to the print menu) was clicked.
         // On the Macintosh, platformPrint is a THPrint; on Windows,
         // platformPrint is a structure (defined in npapi.h) containing
         // the printer name, port, etc.
         //
         void* platformPrint = printInfo->print.fullPrint.platformPrint;
         NPBool printOne = printInfo->print.fullPrint.printOne;

         printInfo->print.fullPrint.pluginPrinted = FALSE; // Do the default

      }
      else   // If not fullscreen, we must be embedded
      {
         //
         // Note: If your plugin is embedded, or is full-screen
         // but you returned false in pluginPrinted above, NPP_Print
         // will be called with mode == NP_EMBED.  The NPWindow
         // in the printInfo gives the location and dimensions of
         // the embedded plugin on the printed page.  On the Macintosh,
         // platformPrint is the printer port; on Windows, platformPrint
         // is the handle to the printing device context.
         //
         NPWindow* printWindow = &(printInfo->print.embedPrint.window);
         void* platformPrint = printInfo->print.embedPrint.platformPrint;

      }
   }
}
```

```
//-------------------------------------------------------------------
// NPP_HandleEvent:
// Mac-only.
//-------------------------------------------------------------------
int16 NPP_HandleEvent(NPP instance, void* event)
{
  NPBool eventHandled = FALSE;
  if (instance == NULL)
    return eventHandled;

  PluginInstance* This = (PluginInstance*) instance->pdata;

  //
  // Note: The "event" passed in is a Macintosh
  // EventRecord*. The event.what field can be any of the
  // normal Mac event types, or one of the following additional
  // types defined in npapi.h: getFocusEvent, loseFocusEvent,
  // adjustCursorEvent. The focus events inform your plugin
  // that it will become, or is no longer, the recepient of
  // key events. If your plugin doesn't want to receive key
  // events, return false when passed at getFocusEvent. The
  // adjustCursorEvent is passed repeatedly when the mouse is
  // over your plugin; if your plugin doesn't want to set the
  // cursor, return false. Handle the standard Mac events as
  // normal. The return value for all standard events is currently
  // ignored except for the key event: for key events, only return
  // true if your plugin has handled that particular key event.
  //

  return eventHandled;
}
```

h:mg:hyperkey:npshell.cpp


```
//*************************************************************
//
//              HyperCD I/O
//
//    Function:
//              Remap-decrypt-decode and merge the
//              missing CRITICAL DATA with the CRIPPLED
//              HyperCD files.
//
//*************************************************************

#include <windows.h>
#include <mmsystem.h>
#include <digitalv.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <string.h>

InsertHyperCDIOModule();
RemoveHyperCDIOModule();
OPEN_HYPERCD();

HDVDCALLBACK HYPERCDIO(LPHYPERCDINFO)
{
  switch (uMessage) {
        case MMIOM_OPEN:
              HYPERCD_Open();
              return 0;

        case MMIOM_CLOSE:
              HYPERCD_Close();
              return 0;

        case MMIOM_READ:
              HYPERCD_ReadMultipleFiles();
              return (LPHYPERCDINFO.displacement);

        case MMIOM_SEEK:
              switch (lParam2) {
                case SEEK_SET:   // seek to the absolute position relative to
original beginning
                        HYPERCD_SeekSet();
                        break;
```

```
                case SEEK_CUR:   // move forward relative to the current
position
                    HYPERCD_SeekCur();

                case SEEK_END:   // seek all the way to the end
                    HYPERCD_SeekEnd();
                    break;
                    }
                return HYPERCD_Offset();

    default:
                return -1;
  }
}h:\word
```


The following is the software listing for encrypting the data on the DVD-ROM and cripples the data files thereon allowing playback only on a DVD player that recognizes the Hyper-DVD nature of the DVD-ROM.

```
//////////////////////////////////////////////////////////
//              cutter.c
//
//   Function; this routine cuts a DVD file into multiple
//        sections and encrypts them onto a DVD rom.
//        Critical section will be removed and stored on remote
//        server. This also alerts DVD player to foreign file
//        format and initiates search for permission keys.
//////////////////////////////////////////////////////////

#include <Windows.h>
#include <stdlib.h>
#include <stdlo.h>
#include <time.h>
#include <string.h>
#include <sys\types.h>
#include <sys\stat.h>

#define MAX HAEDER_SIZE 64000
szBuffer_PGQ[MAX_HAEDER_SIZE];
void flllJunk(char * statBuffer,Int n);

intFileCutter(HWND hWnd,char *fn)
{
     Int V,T,L,nType;
     Int n;
     Int DEBUG =0;
     FILE *pinputFile, *out;
     char filename[MAX_STR];
     char header[MAX_HEADER], *p, msg[MAX_STR];
     char sxNumber[MAX_STR];

     struct_stat statBuffer;
     Int nResult;
     long IRemainBytes;
     long IReadBufferSize;
     long IFileSize;
     long IIndex;
     long IBytesRead.IBytesWrite;

     charszinFile[128],szOutFile[128], *pin,*pOut;
         char szOutfile_PGZ[ ]= BIG.PGQ";
     FILE *out2;

     //get switch -s
     strcpy(filename.fn);
```

```
_strupr(filename);

p=filename;
pin=szinFile;
//first SPACE
whlle(p*= =")
  p ++;
  //copy IN file name
while(*p !="&&*pl=0)
  {
  *pin ++ = *p++;
  }

//skip SPACE
while(*p==")
  p++;
while(*p="&&*pl=O)
  {
  *pout++ = *p++;
  }
*pin = '\O';
*pOut = 'O';

//open files
If(strlen(szinFile)==0)
  {
  MessageBox(NULL, "Please drag&drop the file to me.\r\n
  in DOS,type\"encoder file\".",Encoder V1.2", MB_OK);
  return O;
  }
/*Open file pinputFile bin mode; */
if((pinputFile = fopen (szinFile, "rb")) ==NULL)
  {
  MessageBox(NULL, szinFile, "Error reading file",
  MB_OK);
  return O;
  }

// Now read nd cut the file into many parts;
// Critical part: filename.pgq --> stay on remote server
// Chunky part; filename.pgz --> on local media such as
     DVD-Rom
IBytesRead= fread (szBuffer_PGQ, sizeof(char),
     MAX_HAEDER_SIZE,pinputFlle);
If(out2 = fopen(szOutfile_PGZ, "w+b")) ==NULL)
  {
  MessageBox(NULL,szOutfile_PGZ, "Error creating PGQ
     file", MB_OK);
  return O;
  }
IBytesWRite=fWrite (szBuffer_PGQ, sizeof(char),
     IBytesRead,out2);

if(BytesWrite l+ IBytesRead)
  MessageBox(NULL, "", IBytesWRite l = IBytesRead",
```



```
     MB_OK);
fclose(out2);

// now the chunky part
// file type
if(strstr(szinFile, ".AVI") l = NULL)
  nType = 1;
else if (strstr(szinFile, ".EXE") l = NULL)
  nType = 2;
else if (strstr(szinFile, ".MOV") l = NULL)
  nType = 3;
else if (strstr(szinFile, ".MPG") l = NULL)
  nType = 4;
else if (strstr(szinFile, ".JPG") l = NULL)
  nType = 6;
else if (strstr(szinFile, ".GIF") l = NULL)
  nType = 7;
else if (strstr(szinFile, ".PIC") l = NULL)
  nType = 8;
else if (strstr(szinFile, ".TXT") l = NULL)
  nType = 9;
else if (strstr(szinFile, ".HTM") l = NULL)
  nType = 10;
else if (strstr(szinFile, ".VOB") l = NULL)
  nType = 11;
else
  {

     MessageBox(NULL, "Unrecognizable file", "Encoder
          Error", MB_OK);
     return O;
     }
If(strlen(szOutFile) ==O)
     {
     strcpy(szOutFile, szinFile);
     pOut = strstr(szOutFile,".");
     pOut + +;
     strcpy(pout, "PGZ");
     pOut + =3;
     *pOut=O;

     wsprintf(msg,"Output file not specified,\nNew
          output file: [%s]", szOutFile);
     MessageBox(NULL,msg, "Warningl", MB_OK);

     }

If((out = fopen(szOutFile, "w+b")) == NULL)
     {
     MessageBox(NULL, szOutFile, "Error creating file",
          MB_OK);
     return O;
     }
DEBUGGER(DEBUG,szOutFile, "File createdl");
```

```
//Add HyperLOCK HyperDVD header
AddHyperDVDHeader0;

CutFileIntoMultiplePartsO;

WriteKeyFiles0;
WriteChunkyFiles( );
CloseHDVDFiles0;
}

CutFileIntoMultipleParts( )
{
// Create new data structure to hold
// critical data/keys table/list

pKeyStruct = new HCDKay;
pChunkTable = new HCDChunk;
ExtractKeys (pKeyStruct) pChunkTable);

// now we have keys & chunky data, encryptChunkydata

Encrypt1(pChunkTable, encrypt_method),
Encrypt2(pKeyStruct, en - method2),
// Add encryption method to key structure

Add EMethod ( ) encryp_method);

return;
}
```


The following is the software listing for determining if a standard or Hyper DVD-ROM is to be played by the player, and for seeking the enabling data, trigger or key from a server or a cable-service provider for providing the missing data necessary for the DVD-player to play a Hyper-DVD.

```
//
//              Player.c
//
//    Function:
//                    This file shows a HyperDVD player that
//                    plays back HyperDVD files that have been
//                    encrypted using HyperLOCK patents pending
//                    technologies.
//
//
//


void PlayRegularDVD(void);
GetPermissionFromRemoteServer(int *pPermission);
Display(int Message);
void PlayHyperDVD(void)
{
// HyperDVD files that are protected by using
//      special combinations of county-code and parental
//      control code will not play in regular DVD players.
//      The following function turns on the appropriate
//      switches to allow the files being able to display.
TurnOnHardwareSettings();
PlayDVD();
}

// A power-on switch starts this algorithm
int main(int argc, char **argv)
{
   if(bIsHyperDVDROM()== TRUE)
        {
        GetPermissionFromRemoteServer(&Permitted);
        if(Permitted == TRUE)
                PlayHyperDVD();
        else
                Display(HYPERDVD_WARNING_MESSAGE);

        }
   else
        PlayRegularDVD();

   return 1;
}
```

While specific embodiments of the invention have been shown and described, it is to be understood that numerous changes and modifications may be made therein without departing from the scope, spirit and intent of the invention as set forth in the appended claims.

WHAT I CLAIM IS:

CLAIM 1. A method of transmitting video and/or graphic data files over the Internet or Intranet from a Web site, comprising:

(a) encrypting the video and/or graphic data and storing it at a Web site associated with a server;

(b) encrypting a video player and storing it at the Web site;

(c) downloading the encrypted video and/or graphic data and encrypted video player of said steps (a) and (b) to a requesting computer via the Internet or Intranet;

(d) prior to said step (c), requesting the downloading of said encrypted video and/or graphic data and encrypted video player by the requesting computer;

(e) decrypting the video and/or graphic data and video player at the requesting computer; and

(f) playing back the decrypted video and/or graphic data via the decrypted video player.

CLAIM 2. A method of playing encrypted video and/or graphic data transmitted over the Internet or Intranet from a Web site, comprising:

(a) requesting by an end user's computer the downloading of encrypted video and/or graphic data and an encrypted video player from a Web site of the Internet or Intranet;

(b) receiving the requested encrypted video and/or graphic data and an encrypted video player from the Web site of the Internet or Intranet;

(c) decrypting the encrypted video and/or graphic data and encrypted video player at the requesting computer; and

(f) playing back the decrypted video and/or graphic data via the decrypted video player at the end user's computer.

CLAIM 3. A method of preventing unauthorized use of video and/or graphic data, comprising:

(a) encrypting the video and/or graphic data;

(b) encrypting a video player;

(c) storing at least one of the encrypted video and encrypted player of said steps (a) and (b) at a Web site of the Internet or Intranet;

(d) downloading at least one of the encrypted video and encrypted video player of said steps (a) and (b) to a requesting computer via the Internet or Intranet;

(e) decrypting the encrypted video and/or graphic data and encrypted video player at the requesting computerl; and

(f) playing the decrypted video and/or graphic at the requesting computer via the decrypted player.

CLAIM 4. The method of preventing unauthorized use of video and/or graphic data according to claim 3, wherein said step (c) comprises storing both the encrypted video and encrypted player of said steps (a) and (b) at the Web site of the Internet or Intranet.

CLAIM 5. The method of preventing unauthorized use of video and/or graphic data according to claim 3, wherein said step (c) comprises storing the encrypted player of said step (b) at the Web site of the Internet or Intranet, said step (d)

comprising transmitting the encrypted player to the requesting computer.

CLAIM 6. The method of preventing unauthorized use of video and/or graphic data, according to claim 5, wherein said step (a) comprises storing the encrypted video files at a requesting end-user's computer.

CLAIM 7. The method of preventing unauthorized use of video and/or graphic data, according to claim 3, wherein said step (c) comprises storing the encrypted video and/or graphic data of said step (a) at the Web site of the Internet or Intranet, said step (d) comprising transmitting the encrypted video and/or graphic data to the requesting computer.

CLAIM 8. The method of preventing unauthorized use of video and/or graphic data, according to claim 7, wherein said step (b) comprises storing the encrypted player at a requesting end-user's computer.

CLAIM 9. In a large storage-capacity ROM-disk for storing large amounts of data, such as video and audio, for playback by a player, said ROM-disk having at least one of a parental code means and a country code means thereon, the improvement comprising:

additional code means thereon for preventing playback of said ROM-disk without enabling data.

CLAIM 10. The large storage-capacity ROM-disk for storing large amounts of data according to claim 9, wherein said parental code means comprises one of a first code representing children-only titles that may be played by said player, a second code representing that only adult titles are prevented

from being played by said player, and a third code representing that all titles may be played by said player, wherein said additional code means for preventing playback of said ROM-disk without enabling data comprises a fourth code of said parental code different from said first, second and third codes.

CLAIM 11. The large storage-capacity ROM-disk for storing large amounts of data according to claim 9, wherein said country code means comprises one of a plurality of codes representing a specific country in which said ROM-disk is to be played, said player having a corresponding code matching said one country code allowing playback of said ROM-disk, wherein said additional code means for preventing playback of said ROM-disk without enabling data comprises another unique country code, said another unique country code being one that does not represent an actual country.

CLAIM 12. The large storage-capacity ROM-disk for storing large amounts of data according to claim 9, wherein said ROM-disk is a DVD-ROM disk.

CLAIM 13. The large storage-capacity ROM-disk for storing large amounts of data according to claim 9, in combination with player means for playing back the data on said ROM-disk, said player means comprising differentiating means for differentiating between a ROM-disk having said additional code thereon, and a ROM-disk not having said additional code thereon, whereby when said differentiating means of said player means detects a ROM-disk without said another code

thereon, said player means automatically plays back the data thereon, and whereby if said differentiating means of said player means detects a ROM-disk with said another code thereon, said player means automatically generates a call to a service provider seeking to obtain said enabling data in order to allow playback of said ROM-disk.

CLAIM 14. The large storage-capacity ROM-disk for storing large amounts of data according to claim 13, wherein said player means comprises enabling-data seeking means for calling a service provider for requesting the downloading of said enabling data; said player means further comprising a disk-player, trigger means, and switch means, said trigger means generating a trigger signal in response to the reception of said enabling data from the service provider for actuating said switch means for actuating said disk-player to play the ROM-disk.

CLAIM 15. The large storage-capacity ROM-disk for storing large amounts of data according to claim 14, wherein said ROM-disk comprises encrypted data, said player means further comprising decrypting means for decrypting said data for playback; said trigger means triggering said switch means to couple said decrypting means to said disk-player for decrypting said data in order to allow playback by said disk-player.

CLAIM 16. The large storage-capacity ROM-disk for storing large amounts of data according to claim 14, wherein said player means comprises a microprocessor, and each of said enabling-data seeking means, trigger means, and switch means comprises software code operatively associated with said

microprocessor.

CLAIM 17. The large storage-capacity ROM-disk for storing large amounts of data according to claim 13, wherein said player means comprises enabling-data seeking means for calling a service provider for requesting the downloading of said enabling data; and coupling means coupling said player means to a service provider, said coupling means comprising at least one of a modem for connecting said player means to said service provider, and a cable box for connecting said player means to a cable-TV service provider.

CLAIM 18. The large storage-capacity ROM-disk for storing large amounts of data according to claim 16, wherein said ROM-disk is a DVD-ROM disk.

CLAIM 19. A ROM-disk playing apparatus for discriminating between a large storage-capacity ROM-disk having playback-prevent code means thereon and a ROM-disk not having playback-prevent code means thereon, comprising:

a disk-player for playing back a ROM-disk;

a microprocessor;

memory means for storing software;

software means comprising first means for detecting the presence of a ROM-disk having playback-prevent code means thereon; second means for generating a call to a service provider in response to said first means detecting the presence of said code means, in order to retrieve enabling data for allowing playback of data on a ROM-disk; and third means for generating a trigger to allow said disk-player to playback said data on a ROM-disk.

CLAIM 20. The ROM-disk playing apparatus according to claim 19, wherein said memory means further comprises fourth means for decrypting encrypted data on a ROM-disk; said third means coupling said fourth means for decrypting to said disk-player.

CLAIM 21. The ROM-disk playing apparatus according to claim 20, wherein said disk-player comprises a MPEG-2 video player.

CLAIM 22. The ROM-disk playing apparatus according to claim 19, further comprising coupling means for coupling said second means for generating a call to a service provider, said coupling means comprising at least one of a modem and a cable box.

CLAIM 23. The ROM-disk playing apparatus according to claim 19, in combination with a DVD-ROM disk, said DVD-ROM disk having at least one of a parental code means and a country code means thereon, and playback-prevent code means thereon for preventing playback of said ROM-disk without enabling data;

CLAIM 24. The ROM-disk playing apparatus according to claim 23, wherein said parental code means comprises one of a first code representing children-only titles that may be played by said disk-player, a second code representing that only adult titles are prevented from being played by said disk-player, and a third code representing that all titles may be played by said disk-player, wherein said additional code means for preventing playback of said ROM-disk without enabling data comprises a fourth code of said parental code different from

said first, second and third codes.

CLAIM 25. The ROM-disk playing apparatus according to claim 23, wherein said country code means comprises one of a plurality of codes representing a specific country in which said ROM-disk is to be played, said disk-player having a corresponding code matching said one country code allowing playback of said ROM-disk, wherein said additional code means for preventing playback of said ROM-disk without enabling data comprises another unique country code, said another unique country code being one that does not represent an actual country.

CLAIM 26. A method of labeling a DVD-ROM comprising:

(a) encoding the DVD-ROM with a code that prevents playback of the data on the DVD-ROM without first obtaining enabling data;

(b) said step (a) comprising encoding the DVD-ROM with at least one of a new parental code different from those used for parental control of playback of DVD-titles, and a new country code that does not actually represent a country.

CLAIM 27. A method of playing back a large storage-capacity ROM-disk, comprising:

(a) reading a ROM-disk via a player apparatus, and differentiating between a ROM-disk encoded to prevent playback thereof without enabling data, and a ROM-disk not encoded to prevent playback without enabling data;

(b) playing the ROM-disk if it is not encoded to prevent playback without enabling data;

(c) generating an enabling-data request to a service-

provider if the ROM-disk is encoded to prevent playback without enabling data;

(d) sending the enabling-data request to a service-provider for requesting the return-sending of enabling data that will enable the playback of the ROM-disk with code to prevent playback;

(e) receiving the enabling data from the service provider; and

(f) enabling the playback of the ROM-disk with code to prevent playback by the player apparatus in response to said step (e).

CLAIM 28. The method of playing back a large storage-capacity ROM-disk according to claim 27, wherein before said step (a):

(g) encoding a ROM-disk to prevent playback without having first obtained enabling data therefor.

CLAIM 29. The method of playing back a large storage-capacity ROM-disk according to claim 27, wherein said step (d) comprises communicating with a service provider by at least one of the Internet and a cable-box.

CLAIM 30. The method of playing back a large storage-capacity ROM-disk according to claim 27, wherein said step (f) comprises connecting a decryption means to the player apparatus for decrypting the encrypted data on the ROM-disk.

CLAIM 31. The method of playing back a large storage-capacity ROM-disk according to claim 27, wherein said step (a) comprises reading a DVD disk.

CLAIM 32. A method of transmitting data invoking a crippled

file on a storage medium containing video and/or audio over a network, comprising:

(a) converting analogue video and/or audio data into digital data;

(b) storing the digital data representing the video and/or audio on a storage medium for use by an end user's computer;

(c) crippling the video and/or audio files on the storage medium , whereupon the files are rendered unplayable without an uncrippling trigger;

(d) storing uncrippling trigger data at a host computer for use in uncrippling the data files on the storage medium;

(e) transmitting the uncrippling trigger data from the host computer through a network to the end-user's computer with which the storage medium having the crippled data files thereon is associated;

(f) receiving the uncrippling trigger data at the end-user's computer in the volatile RAM of the end-user's computer; and

(g) instantly uncrippling and playing the crippled data files on the storage medium by means of combining the uncrippling trigger data sent by the host computer in said step (e) with the crippled data on the storage medium.

CLAIM 33. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 32, wherein said step (f) comprises catching the uncrippling trigger data the crippled data files and directing the encoded text format data to a specific directory-location of the end-user computer.

CLAIM 34. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 32, wherein said step (c) comprises removing the header data from the video/audio files; said step (d) comprising storing the header data representing the header data removed from the video/audio files in said step (c).

CLAIM 35. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 32, wherein before said step (e), encoding the uncrippling trigger data from binary format into encoded text format data; and after said step (f), decoding the encoded text format data back into binary format.

CLAIM 36. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 32, wherein said step (g) is carried out immediately after said step (f), and immediately after said step (g), playing the video and/or audio on a player.

CLAIM 37. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 36, wherein said step (f) comprises directing the incoming uncrippling trigger data to a cache directory; said step (g) being performed while said uncrippling trigger data is in said cache directory for immediate playback of said video and/or audio files on said

storage medium.

CLAIM 38. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 35, wherein said step of encoding the uncrippling trigger data from binary format into encoded text comprises converting the binary data into seven-digit ASCII code.

CLAIM 39. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 36, said step of playing comprising converting the digital binary data back into analogue.

CLAIM 40. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 32, wherein said storage medium comprises memory means for representing the necessary information for automatically and directly connecting via the Internet the end-user's computer, with which the storage medium is associated, to a host computer which stores the uncrippling trigger data for the video/audio files on the storage medium .

CLAIM 41. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 32, further comprising:

(h) allowing, by server-permission only, the end-user the ability to store said trigger on non-volatile media for permanent ownership of said data.

CLAIM 42. A method of transmitting data invoking a crippled

file on a storage medium containing video and/or audio data over the Internet, comprising:

(a) storing uncrippling trigger data at a host computer for use in uncrippling video/audio files on a storage medium ;

(b) transmitting the uncrippling trigger data from the host computer through the Internet to the end-user's computer with which the storage medium having the crippled files thereon is associated;

(c) receiving the uncrippling trigger data at the end-user's computer over the Internet; and

(d) uncrippling the crippled data files on the storage medium by means of the uncrippling trigger data sent by the host computer in said step (b).

CLAIM 43. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio data over the Internet, according to claim 42, wherein said step (c) comprises catching the uncrippling trigger data for the crippled data files and directing the encoded text format data to a specific cache-directory location of the end-user computer for immediate playback of the video and/or audio data.

CLAIM 44. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio data over the Internet, according to claim 42, wherein before said step (a), removing the header data from the video/audio files; said step (d) comprising restoring the header data representing the header data removed from the video/audio files.

CLAIM 45. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio data over the Internet, according to claim 42, wherein before said step (b), encoding the uncrippling trigger data from binary format into encoded text format data; and after said step (c), decoding the encoded text format data back into binary format.

CLAIM 46. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio data over the Internet, according to claim 42, wherein after said steps (c) and (d) are carried substantially simultaneously so that is carried out immediately so that the video and/or audio data may be played back substantially immediately after said step (d).

CLAIM 47. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 43, wherein said step (c) comprises directing the incoming uncrippling trigger data to a cache directory.

CLAIM 48. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 42, wherein said storage medium comprises memory means for representing the necessary information for automatically and directly connecting via the Internet the end-user's computer, said method further comprising before said step (a), automatically and directly connecting the end user's computer to the host

computer which has stored thereat the uncrippling trigger data for the video/audio files on the storage medium by means of the memory means of the storage medium for representing the necessary information for automatically and directly connecting via the Internet.

CLAIM 49. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 42, wherein said step (a) comprises storing at least one of the following: Video/audio header data; data for removing the hidden-status flag for the video/audio data files on the storage medium; data for unzipping the zipped data files of the video/audio data files on storage medium; data for changing the extension of the video/audio data files.

CLAIM 50. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 42, wherein said storage medium comprises at least one of: CD-ROM, floppy disk, and hard drive.

CLAIM 51. The method of transmitting data invoking a crippled file on a storage medium containing video and/or audio over the Internet, according to claim 47, further comprising permanently storing the incoming uncrippling trigger data in ROM of the end-user's computer, for repeatedly uncrippling the crippled file when the playing thereof is required.

CLAIM 52. In a CD-ROM for use with a computer, which CD-ROM Internet comprises memory means for storing binary data thereon, the improvement comprising:

said memory means containing data files representative of video and/or audio;

said data files being crippled, whereby, without uncrippling trigger data, said data files are not capable of being played by a computer.

CLAIM 53. The CD-ROM for use with a computer according to claim 52, wherein said crippled data files lack the necessary audio/video header information.

CLAIM 54. The CD-ROM for use with a computer according to claim 52, in combination with an end-user's computer for use in playing back the data files on the CD-ROM, a host computer having a memory storing said uncrippling data thereon, and the Internet system linking said end-user's computer with said host computer, whereupon said host computer's sending said uncrippling data stored in said memory thereof to said end-user's computer, said crippled data files on said CD-ROM associated with said end-user's computer is uncrippled and rendered playable.

CLAIM 55. A method of transmitting data over the Internet invoking a crippled file contained on a storage medium containing stored, crippled digital-data information, comprising:

(a) storing uncrippling trigger data at a host computer for use in uncrippling the data files on the storage medium;

(b) transmitting the uncrippling trigger data from the host computer's server through the Internet to the end-user's computer with which the storage medium having the crippled data files thereon is associated;

(c) receiving the uncrippling trigger data at the end-user's computer; and

(d) uncrippling the crippled data files on the storage medium by means of the uncrippling trigger data sent by the host computer in said step (b).

CLAIM 56. A method of receiving triggering data for a crippled file at a receiving computer over the Internet, comprising:

(a) establishing a socket-to-socket connection between a host computer, from which the video and/or audio trigger data is being transmitted on the Internet, and a receiving computer or terminal;

(b) receiving the trigger data over the Internet at the receiving computer or terminal, said trigger data allowing the uncrippling of the vide and/or audio files stored at the receiving computer;

(c) catching the trigger data at the receiving computer or terminal, and directing the trigger data to a specific directory location in computer memory of the receiving computer or terminal;

(d) decoding the trigger data into binary format, and, thereafter;

(e) playing the video and/or audio files stored at the receiving computer.

CLAIM 57. The method of receiving triggering data for a crippled file according to claim 56, wherein said step (c) comprises directing the incoming encoded text format data to

a RAM cache directory.

CLAIM 58. An apparatus for receiving de-crippling video and/or audio data over the Internet at a receiving computer or terminal, comprising:

a computer means comprising at least one memory means for fixedly storing information, said memory means storing crippled video and/or audio files;

means for establishing a point-to-point connection between the computer means and a host computer from which the de-crippling video and/or audio data is being transmitted on the Internet, and for receiving the de-crippling video and/or audio data over the Internet;

said memory means of said computer means further comprising software means for catching the de-crippling video and/or audio data and directing it to a specific directory-location in the RAM of said computer means, and for directing the de-crippling video and/or audio data to a player means for the playing thereof;

said computer means further comprising player means for playing the video and/or audio, said de-crippling video and/or audio data thereby allowing the playing of the video and/or audio files stored on said memory means.

CLAIM 59. The apparatus for receiving de-crippling video and/or audio data over the Internet at a receiving computer or terminal, according to claim 58, wherein said software means for catching the de-crippling video and/or audio data directs the de-crippling video and/or audio data to a cache-directory of RAM.

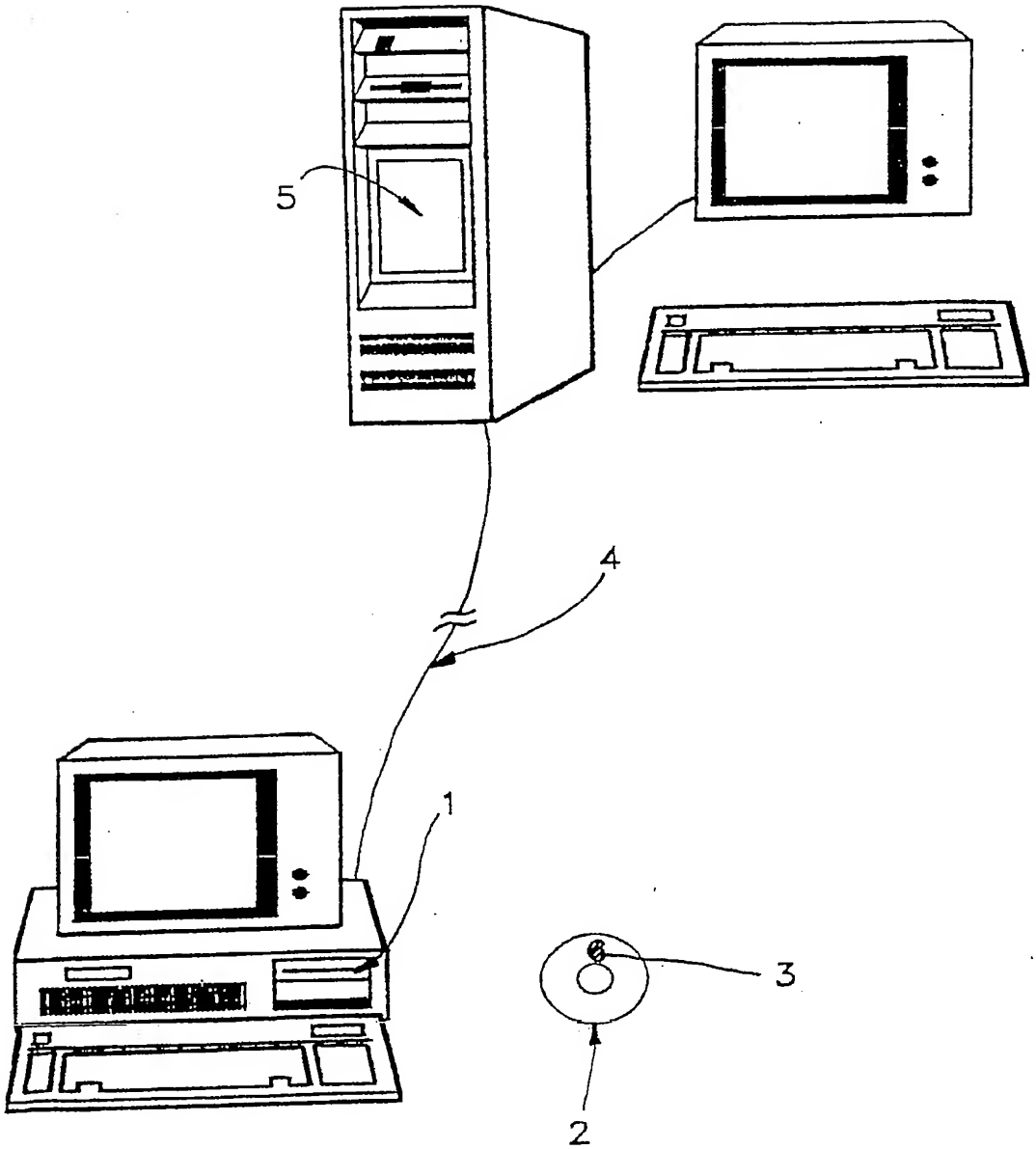

Fig 1

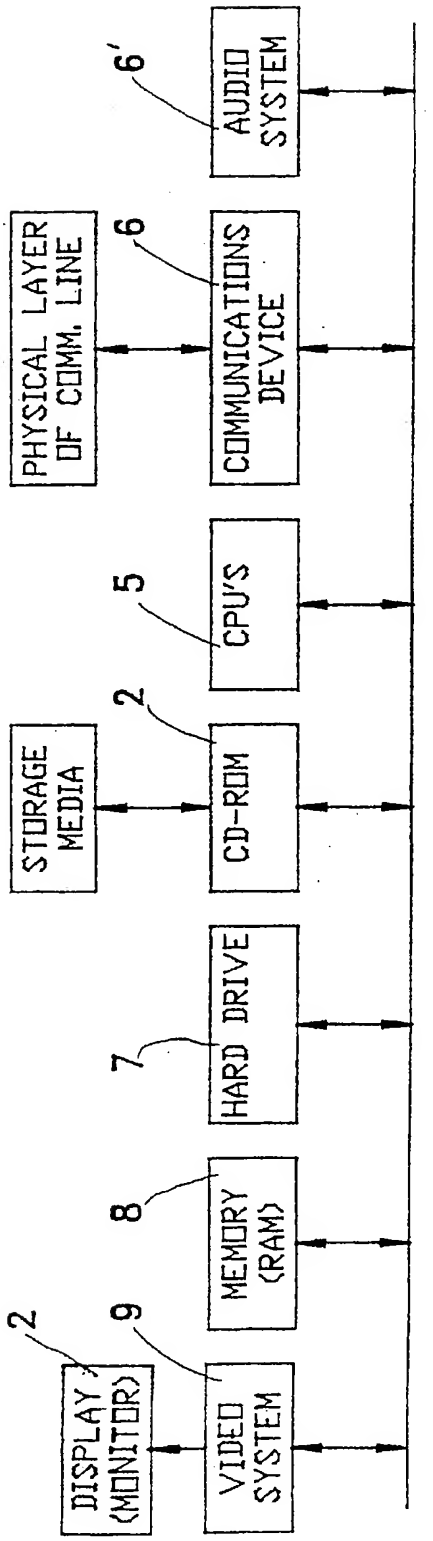
DISPLAY (MONITOR)
2
VIDEO SYSTEM
9
MEMORY (RAM)
8
HARD DRIVE
7
CD-ROM
2
STORAGE MEDIA
CPU'S
5
COMMUNICATIONS DEVICE
6
PHYSICAL LAYER OF COMM. LINE
AUDIO SYSTEM
6'

FIG 2

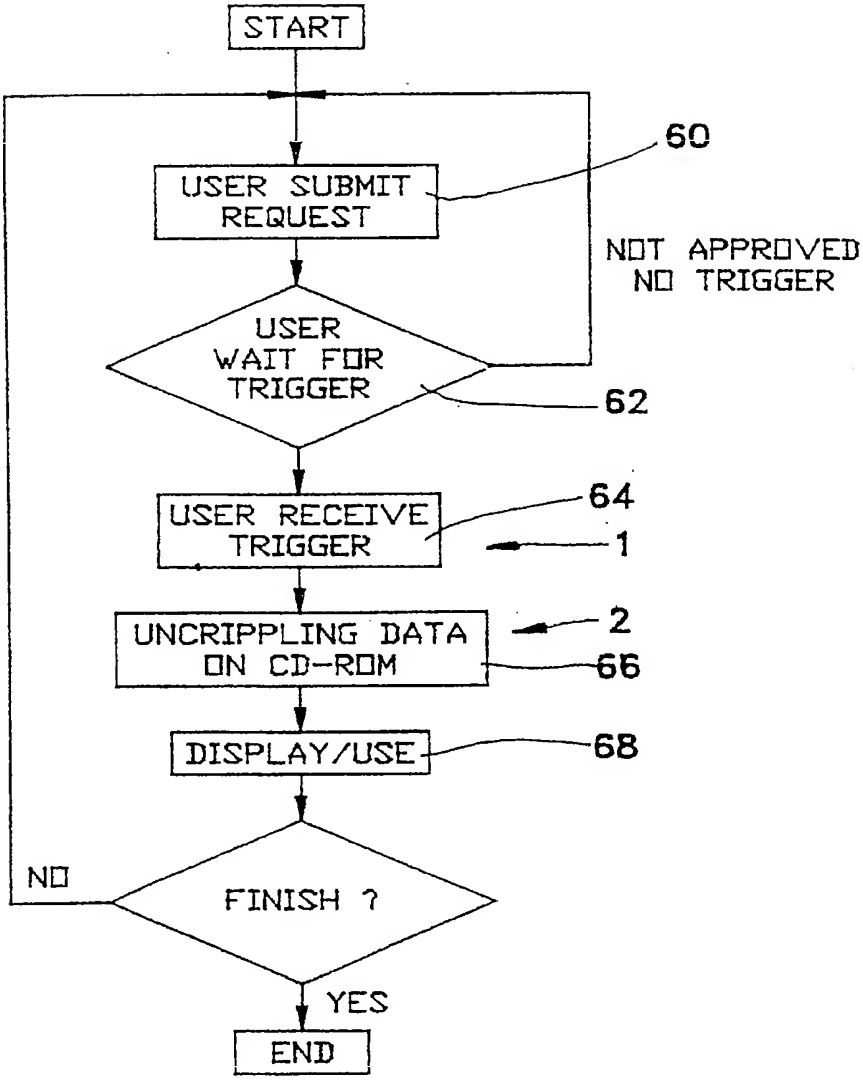
START
USER SUBMIT REQUEST
60
NOT APPROVED NO TRIGGER
USER WAIT FOR TRIGGER
62
USER RECEIVE TRIGGER
64
1
UNCRIPPLING DATA ON CD-ROM
2
66
DISPLAY/USE
68
NO
FINISH ?
YES
END

FIG 3

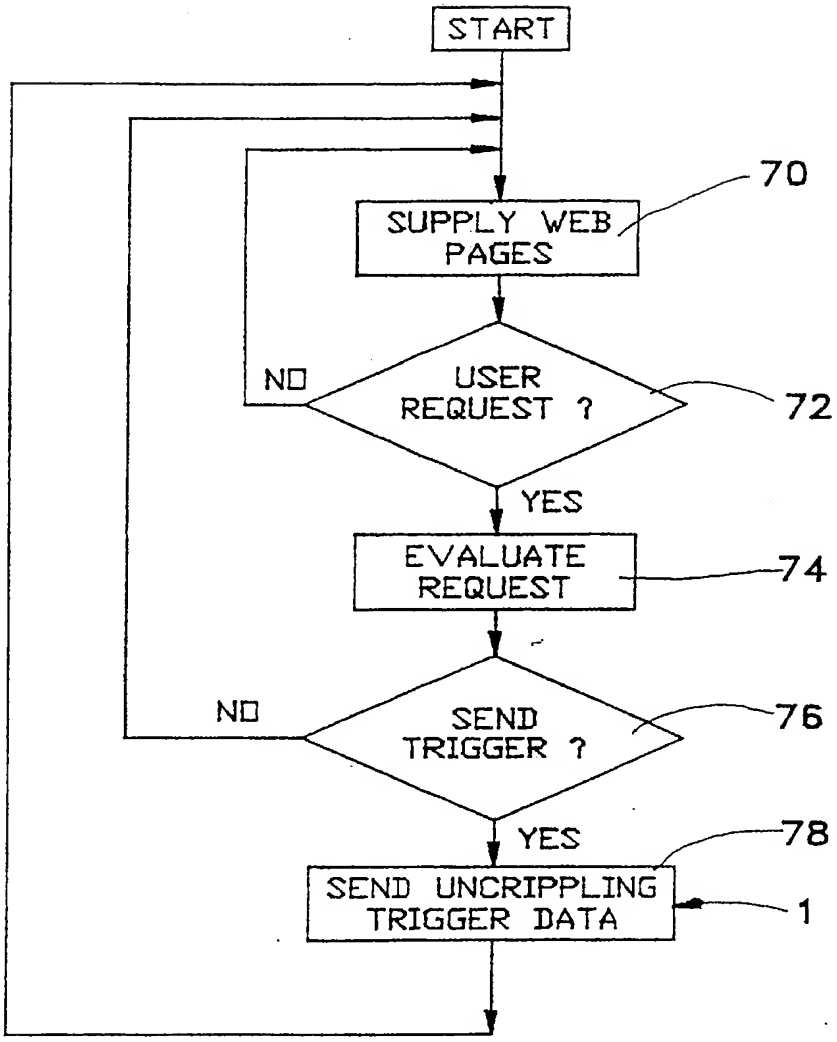
START
SUPPLY WEB PAGES
70
NO
USER REQUEST ?
72
YES
EVALUATE REQUEST
74
NO
SEND TRIGGER ?
76
YES
78
SEND UNCRIPPLING TRIGGER DATA
1

FIG 4

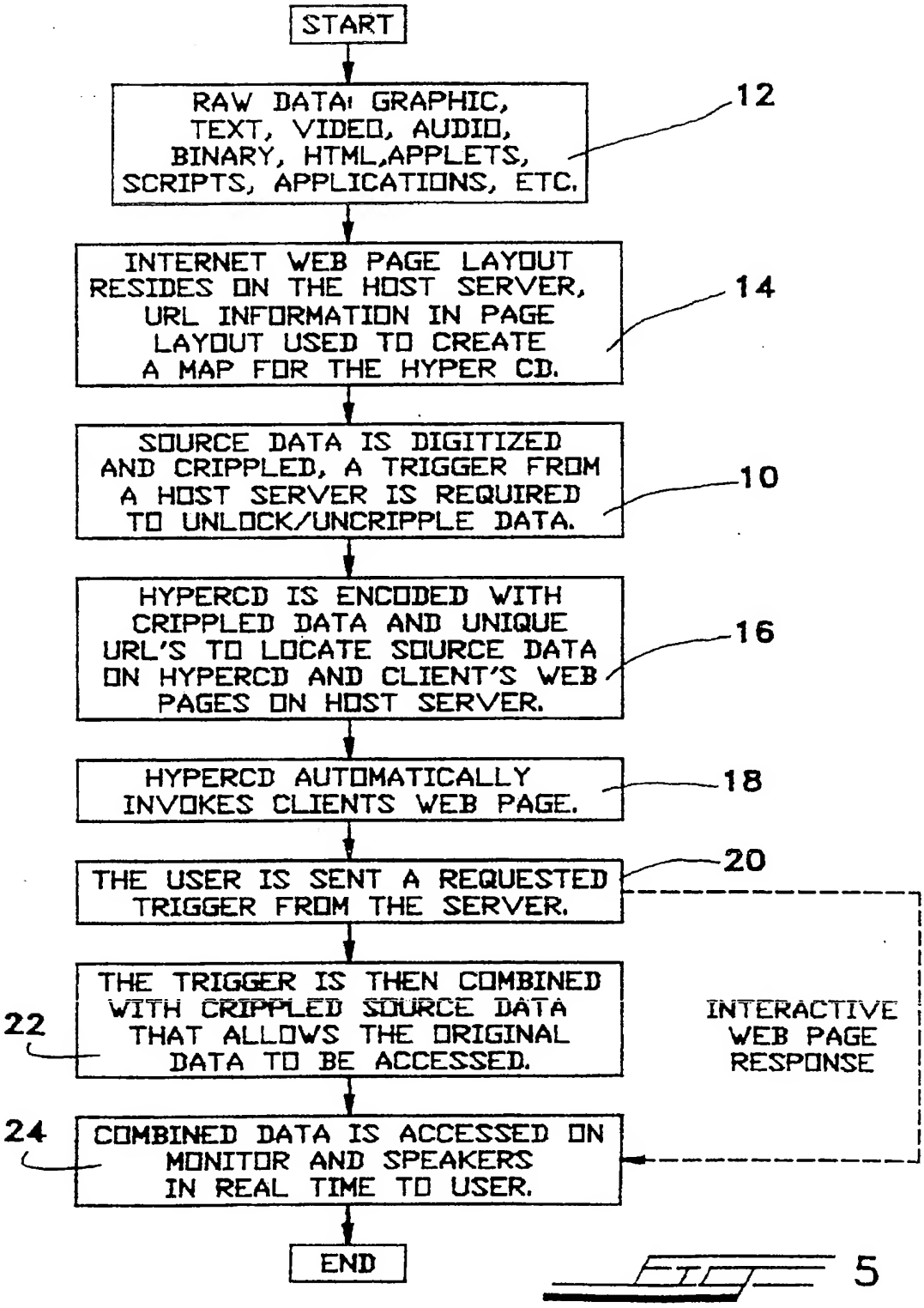
START
RAW DATA: GRAPHIC, TEXT, VIDEO, AUDIO, BINARY, HTML,APPLETS, SCRIPTS, APPLICATIONS, ETC.
12
INTERNET WEB PAGE LAYOUT RESIDES ON THE HOST SERVER, URL INFORMATION IN PAGE LAYOUT USED TO CREATE A MAP FOR THE HYPER CD.
14
SOURCE DATA IS DIGITIZED AND CRIPPLED, A TRIGGER FROM A HOST SERVER IS REQUIRED TO UNLOCK/UNCRIPPLE DATA.
10
HYPERCD IS ENCODED WITH CRIPPLED DATA AND UNIQUE URL'S TO LOCATE SOURCE DATA ON HYPERCD AND CLIENT'S WEB PAGES ON HOST SERVER.
16
HYPERCD AUTOMATICALLY INVOKES CLIENTS WEB PAGE.
18
THE USER IS SENT A REQUESTED TRIGGER FROM THE SERVER.
20
THE TRIGGER IS THEN COMBINED WITH CRIPPLED SOURCE DATA THAT ALLOWS THE ORIGINAL DATA TO BE ACCESSED.
22
COMBINED DATA IS ACCESSED ON MONITOR AND SPEAKERS IN REAL TIME TO USER.
24
INTERACTIVE WEB PAGE RESPONSE
END

FIG 5

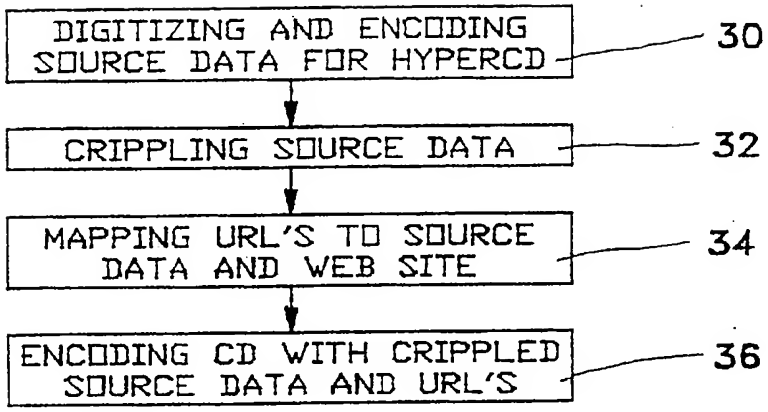
DIGITIZING AND ENCODING SOURCE DATA FOR HYPERCD
30
CRIPPLING SOURCE DATA
32
MAPPING URL'S TO SOURCE DATA AND WEB SITE
34
ENCODING CD WITH CRIPPLED SOURCE DATA AND URL'S
36

FIG 6

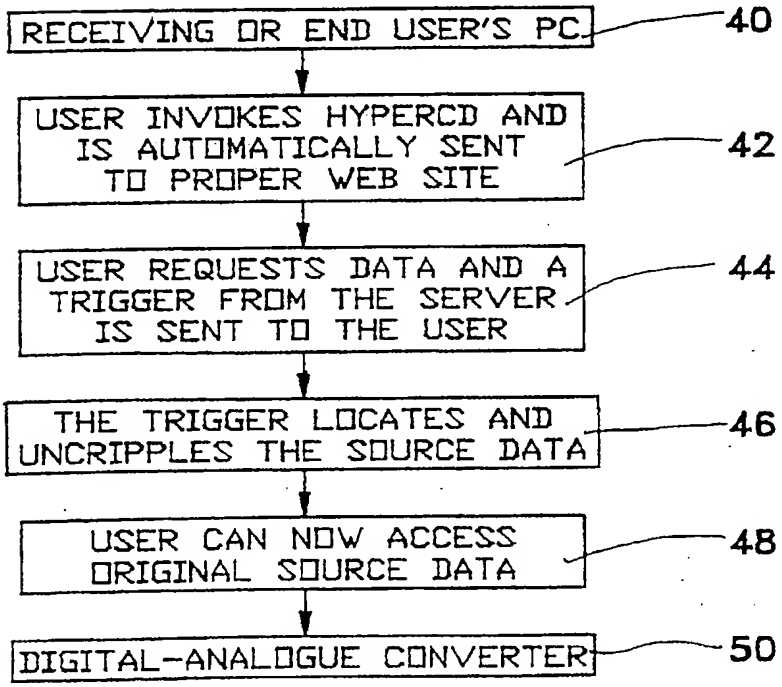
RECEIVING OR END USER'S PC
40
USER INVOKES HYPERCD AND IS AUTOMATICALLY SENT TO PROPER WEB SITE
42
USER REQUESTS DATA AND A TRIGGER FROM THE SERVER IS SENT TO THE USER
44
THE TRIGGER LOCATES AND UNCRIPPLES THE SOURCE DATA
46
USER CAN NOW ACCESS ORIGINAL SOURCE DATA
48
DIGITAL-ANALOGUE CONVERTER
50

FIG 7

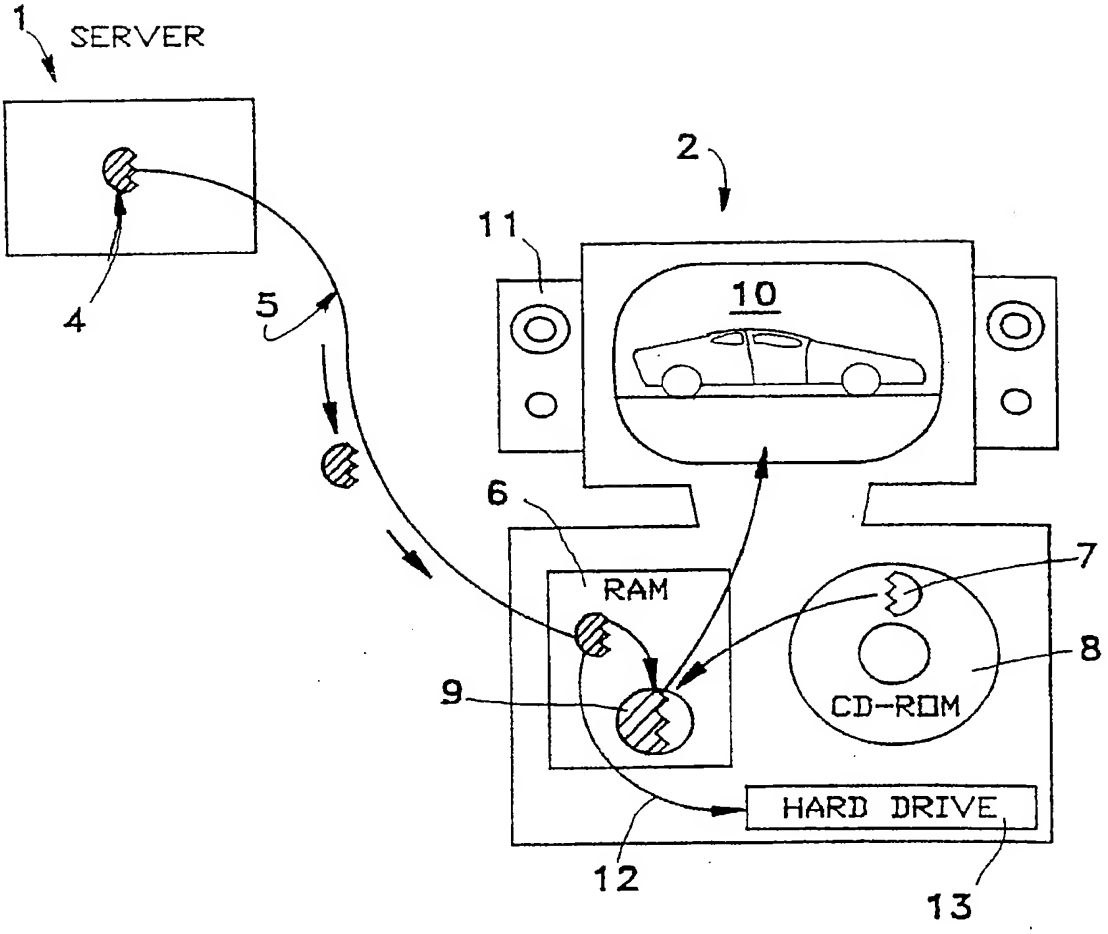

FIG 8

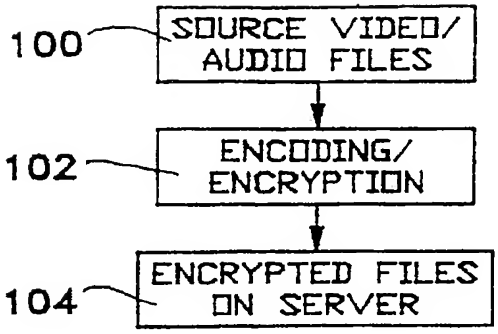
100
SOURCE VIDEO/ AUDIO FILES
102
ENCODING/ ENCRYPTION
104
ENCRYPTED FILES ON SERVER

FIG 9

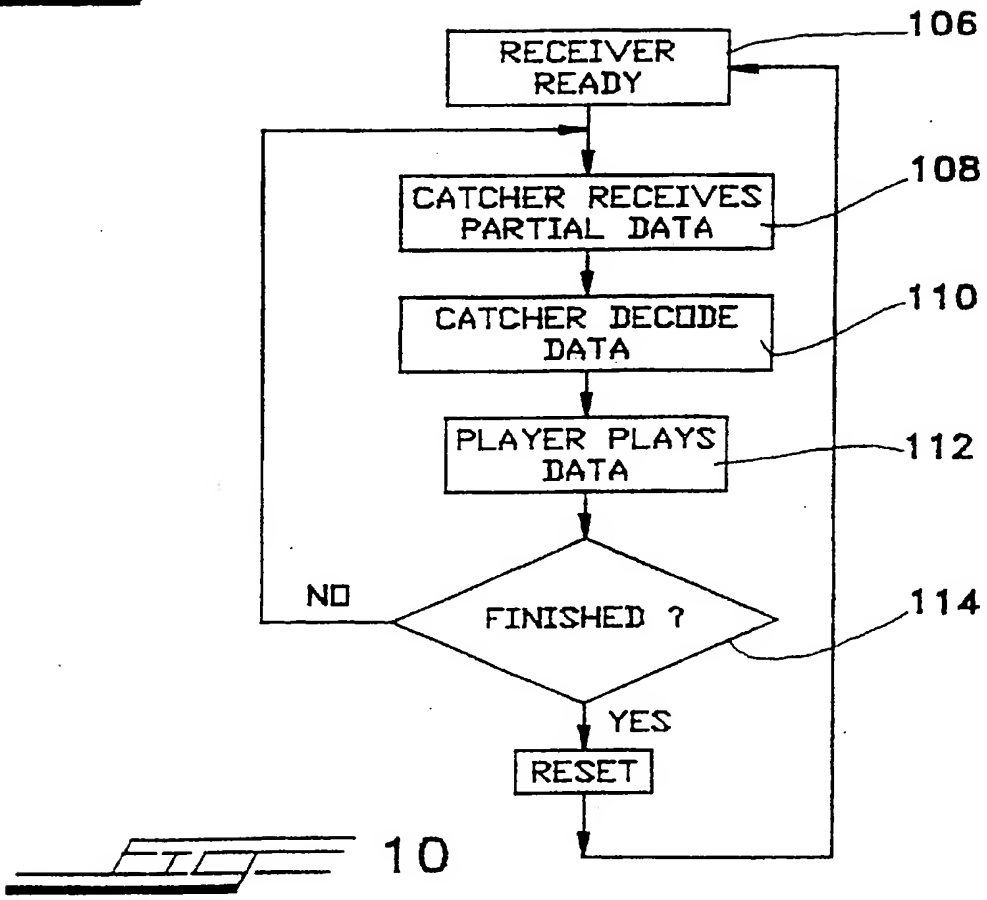
106
RECEIVER READY
108
CATCHER RECEIVES PARTIAL DATA
110
CATCHER DECODE DATA
112
PLAYER PLAYS DATA
NO
FINISHED ?
114
YES
RESET

FIG 10

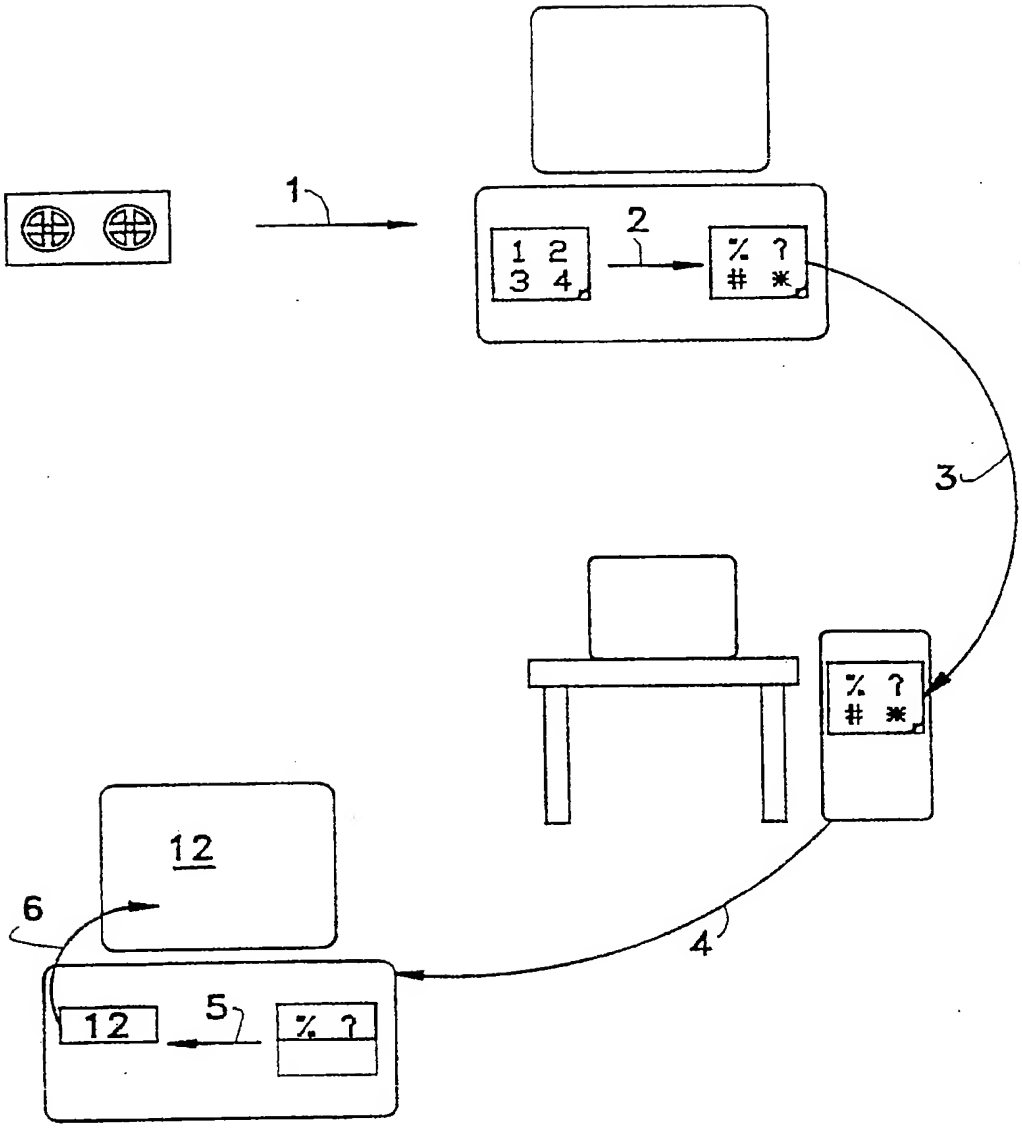

FIG 11

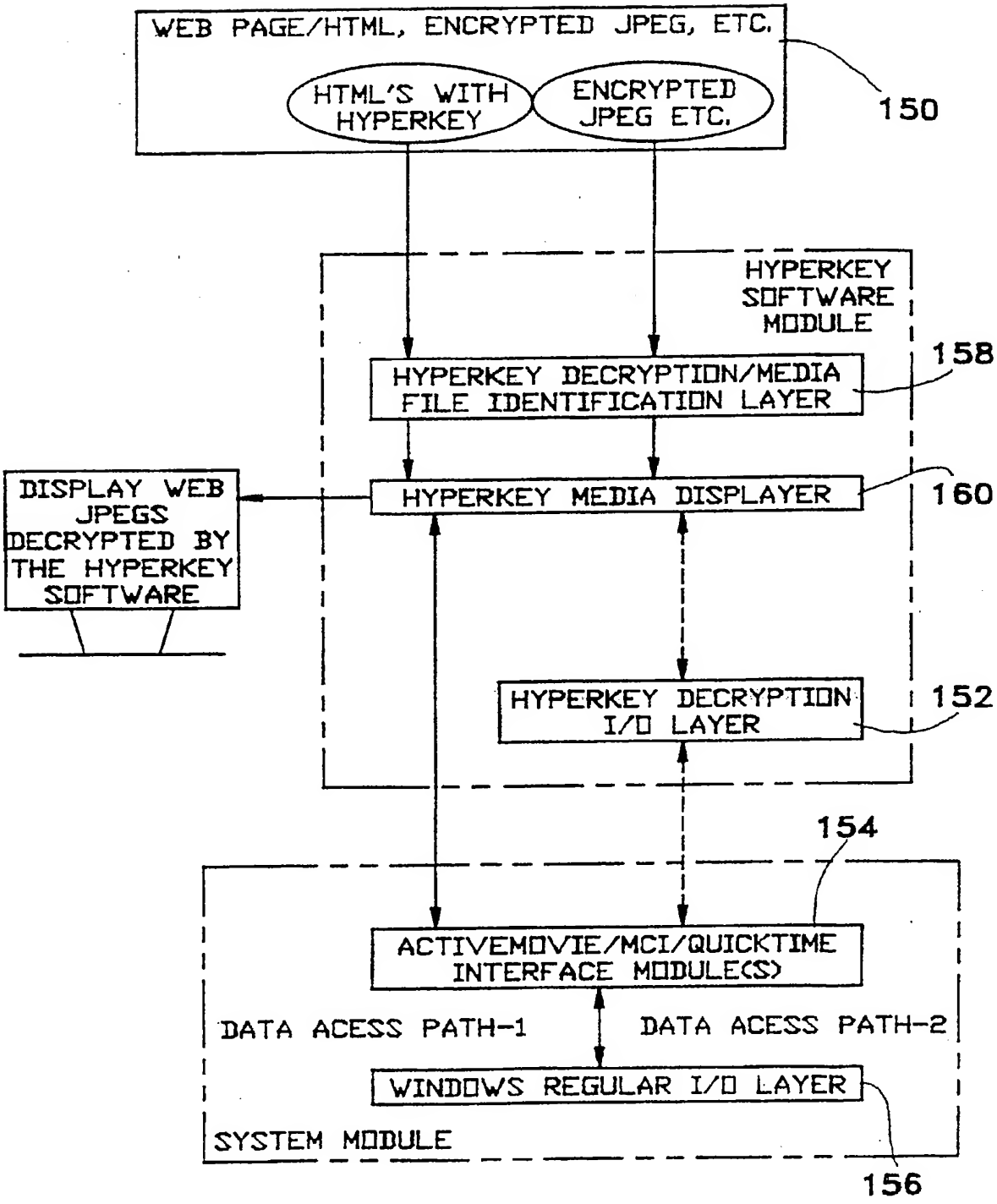
WEB PAGE/HTML, ENCRYPTED JPEG, ETC.
HTML'S WITH HYPERKEY
ENCRYPTED JPEG ETC.
150
HYPERKEY SOFTWARE MODULE
HYPERKEY DECRYPTION/MEDIA FILE IDENTIFICATION LAYER
158
HYPERKEY MEDIA DISPLAYER
160
DISPLAY WEB JPEGS DECRYPTED BY THE HYPERKEY SOFTWARE
HYPERKEY DECRYPTION I/O LAYER
152
154
ACTIVEMOVIE/MCI/QUICKTIME INTERFACE MODULE(S)
DATA ACESS PATH-1
DATA ACESS PATH-2
WINDOWS REGULAR I/O LAYER
SYSTEM MODULE
156

FIG 12

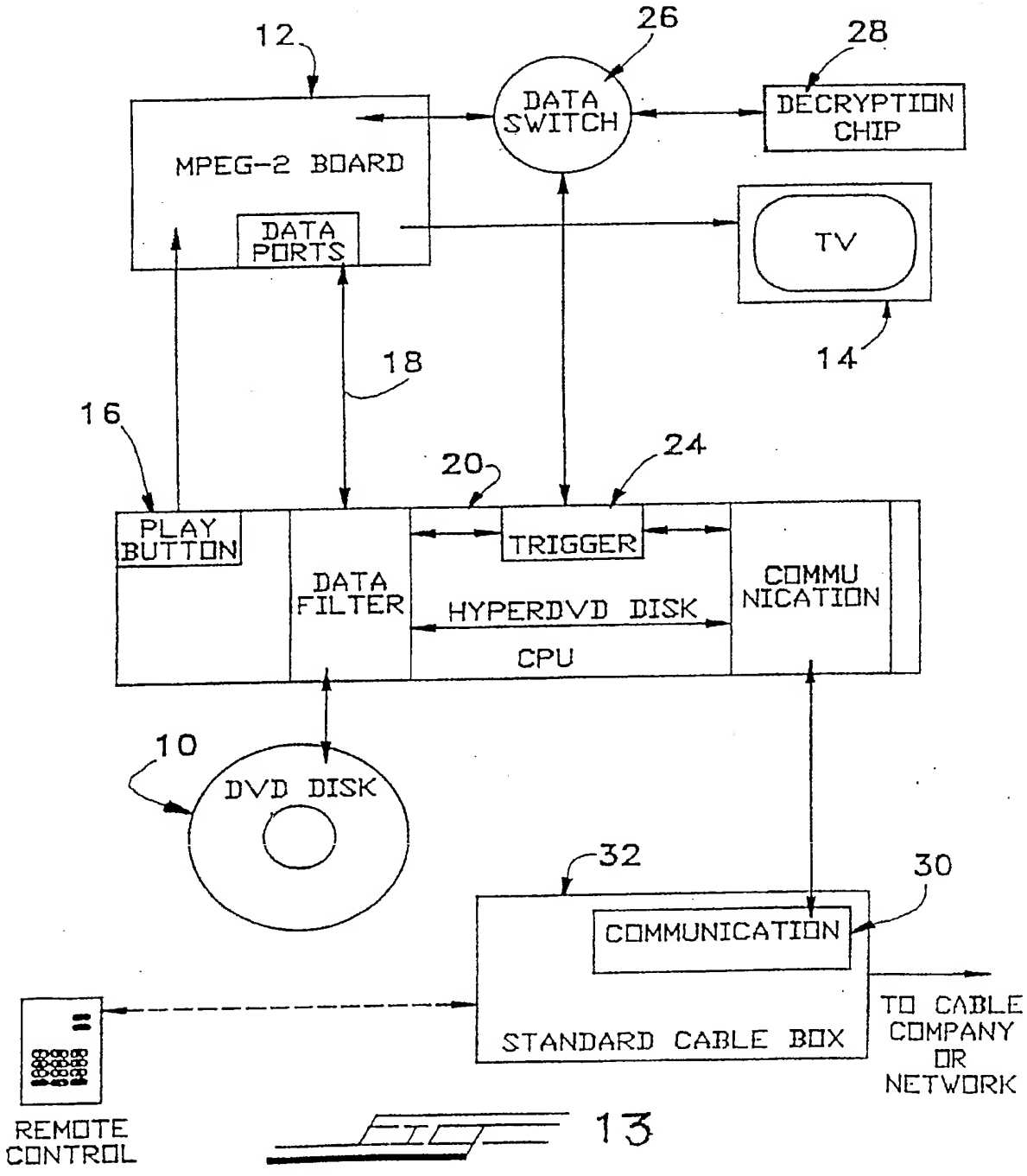
12
26
28
DATA SWITCH
DECRYPTION CHIP
MPEG-2 BOARD
DATA PORTS
TV
14
18
16
20
24
PLAY BUTTON
TRIGGER
DATA FILTER
HYPERDVD DISK
CPU
COMMU NICATION
10
DVD DISK
32
30
COMMUNICATION
STANDARD CABLE BOX
TO CABLE COMPANY OR NETWORK
REMOTE CONTROL

FIG 13